UNIVERSIDADE FEDERAL DO ESTADO DO RIO DE JANEIRO
CENTRO DE CIÊNCIAS EXATAS E TECNOLOGIA
ESCOLA DE INFORMÁTICA APLICADA

Computação em Nuvem: Um Estudo de caso para a Implementação no CCET

Tiago Ribeiro da Silva Melo
Lucas Guilherme de Azevedo

Orientador
Morganna Carmen Diniz

RIO DE JANEIRO, RJ – BRASIL
DEZEMBRO DE 2019

Catalogação informatizada pelo autor

M528

Melo, Tiago Ribeiro da Silva
Computação em nuvem: um estudo de caso para a implementação no CCET / Tiago Ribeiro da Silva Melo. -- Rio de Janeiro, 2019.
55 f.

Orientadora: Morganna Carmen Diniz.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2019.

1. Computação em Nuvem. 2. Infraestrutura. 3. OpenStack. 4. Estudo de Caso. I. Diniz, Morganna Carmen, orient. II. Título.

A994

Azevedo, Lucas Guilherme de
Computação em nuvem: um estudo de caso para a implementação no CCET / Lucas Guilherme de Azevedo. -- Rio de Janeiro, 2019.
55 f.

Orientadora: Morganna Carmen Diniz.
Trabalho de Conclusão de Curso (Graduação) - Universidade Federal do Estado do Rio de Janeiro, Graduação em Sistemas de Informação, 2019.

1. Computação em Nuvem. 2. Infraestrutura. 3. OpenStack. 4. Estudo de Caso. I. Diniz, Morganna Carmen, orient. II. Título.

Computação em Nuvem: Um Estudo de caso no CCET

Tiago Ribeiro da Silva Melo
Lucas Guilherme de Azevedo

Projeto de Graduação apresentado à Escola de Informática Aplicada da Universidade Federal do Estado do Rio de Janeiro (UNIRIO) para obtenção do título de Bacharel em Sistemas de Informação.

Aprovado por:

_____

DSc. Morganna Carmen Diniz (UNIRIO)

_____

DSc. Sidney Cunha de Lucena

_____

DSc. Leonardo Luiz Alencastro Rocha

RIO DE JANEIRO, RJ – BRASIL.

DEZEMBRO DE 2019

Agradecimentos

Agradeço primeiramente à Deus pela oportunidade de ter vivenciado uma das melhores experiências da minha vida.

Aos meus pais por todo suporte e apoio, me incentivando sempre a superar os desafios e a encarar a vida como uma oportunidade de aprendizado, me dando toda o suporte principalmente emocional para que eu pudesse prosseguir e superar os obstáculos propostos pela vida me dando confiança e força. E também aos meus irmãos por toda parceria nos momentos de angústia.

À minha companheira Ana Carolina por todo o carinho, amor, amizade e paciência ao longo destes 5 anos juntos, sendo muitas vezes o ombro em momentos às vezes complicados e difíceis.

Aos meus colegas da UNIRIO que me ajudaram nos trabalhos, nos desabafos e nos estudos, sendo alguns amigos que levarei para a vida toda.

À minha orientadora Morganna que confiou em mim e no Lucas na construção e no desenvolvimento deste trabalho, orientando de forma clara e concisa nos processos e andamento da pesquisa.

E uma gratidão especial ao time com o qual trabalho que possibilitou que eu tivesse experiências essenciais para meu crescimento profissional e intelectual fazendo com que eu elevasse todos os conceitos adquiridos na UNIRIO durante esses anos e estimulando a busca constante por novos aprendizados e tecnologias, como a apresentada neste trabalho.

- Tiago Ribeiro da Silva Melo

Começo agradecendo à equipe por trás do curso de BSI da UNIRIO, desde professores à equipe da secretaria, todos eles que são vitais para manter a boa qualidade e bom funcionamento do curso como um todo, o que me fez valorizar a oportunidade em mãos e incentivando a não desistir em momentos difíceis da minha formação.

À também muitos colegas que se tornaram grandes amigos durante minha formação, cujo compartilhei de experiências boas e ruins, sempre nos motivando a seguir em frente, fazendo essa jornada mais marcante e enriquecedora.

À equipe com que trabalho profissionalmente, que me acompanhou por dois anos de estágio e mais dois anos de efetivo durante a minha formação na UNIRIO, sempre sendo compreensíveis e me ajudando como possível para poder finalizar minha formação, enquanto valorizando a mesma.

À minha companheira que me apoiou e me suportou em momentos ruins e comemorou comigo momentos de vitória, acompanhando meu crescimento pessoal ao longo da jornada.

E, principalmente, agradeço a meus pais, cujas histórias de vida com origens muito humildes sempre foram inspiração para mim, responsáveis por me prover todas as ferramentas necessárias para o descobrimento da minha vocação e indo além para me proporcionar o melhor para que eu atingisse meus objetivos. A confiança que meus pais sempre tiveram no meu potencial foi, e ainda é, essencial para mim.

- Lucas Guilherme de Azevedo

RESUMO

Este trabalho apresenta um estudo de caso sobre o uso de computação em nuvem no CCET. A partir da análise dos serviços oferecidos pelo NTI (Núcleo de Tecnologia da Informação) verificou-se a possibilidade de fornecer um serviço cujo objetivo era democratizar e uniformizar o uso e o acesso a infraestrutura de TI (Tecnologia da Informação) atual sem que houvessem gastos excessivos com orçamentos e aquisições de *software* e *hardware*. Para isso, foi então utilizada a tecnologia de Computação em Nuvem, que possibilita o reuso desta infraestrutura levando independência e autonomia ao usuário final no gerenciamento de recursos.

Para construção desta arquitetura foi escolhida a ferramenta de gerenciamento de nuvem Openstack para abrigar estes serviços do NTI como *moodle* e *DNS* considerando requisitos como custo, segurança, disponibilidade e governança. Com este trabalho, espera-se que os usuários adquiram conhecimento sobre algumas das possibilidades oferecidas pela Computação em Nuvem, vantagens e desvantagens de adoção deste tipo de tecnologia e das formas de gerenciamento de infraestrutura utilizando uma plataforma de gerenciamento distribuída.

Sendo assim, esta pesquisa tem o objetivo de propor uma arquitetura baseada na ferramenta de gerenciamento de Nuvem OpenStack, levando em consideração fatores como segurança, cultura, perfis de usuário diferentes, mantendo o máximo de confiabilidade e disponibilidade dos serviços já presentes, fazendo com que alunos, professores e servidores não só utilizem melhor os recursos tecnológicos do CCET, mas também o utilizem para criação e desenvolvimento de novas tecnologias.

Palavras-chave: computação em nuvem, infraestrutura, estudo de caso, OpenStack.

ABSTRACT

This paper presents a case study on the use of cloud computing at CCET. From the analysis of the services offered by NTI (Information Technology Nucleus), the possibility of offering a service that democratized the use of the current IT (Information Technology) infrastructure was verified without excessive expenses with budgets and software acquisitions. and hardware. For that, the Cloud Computing technology was used, which allows the reuse of this infrastructure, leading independence and autonomy to the end user in the management of own resources.

To build this architecture, the cloud management tool Openstack was chosen to maintain the services already performed by NTI such as moodle and DNS considering requirements such as cost, security, availability and governance. With this work, it is expected that users acquire knowledge about the possibilities offered by Cloud Computing, advantages and disadvantages of adopting this type of technology and the forms of infrastructure management using an open source platform.

Therefore, this research aims to propose an architecture based on the OpenStack Cloud management tool, taking into account the implementation risks and maintaining the maximum reliability and availability of the services already present.

Keywords: cloud computing, infraestrutura,case study, OpenStack.

Índice

Índice de Tabela

Índice de Figuras

1 INTRODUÇÃO

1.1 Motivação

O surgimento da computação em nuvem, também chamada de *cloud computing*, veio para ser um agente de significativas mudanças tanto no âmbito da infraestrutura quanto no desenvolvimento de software.

A UNIRIO apesar de utilizar o serviço público do *google* para domínio de *email* não possui uma arquitetura de nuvem própria servindo de insumo para os serviços fornecidos para os discentes, os docentes e os técnicos administrativos. Atualmente, os serviços essenciais e os projetos acadêmicos são implementados em servidores virtualizados.

As vantagens da adoção da tecnologia proposta remetem à modernização do parque tecnológico com baixo custo de manutenção, de suporte e de investimentos em compra de software. Além disso, há vantagens como um melhor nível de abstração da aplicação em relação à infraestrutura, o uso de recursos de provisionamento sob demanda de forma distribuída ou local. Outras características dessa tecnologia são autonomia e flexibilização na utilização dos recursos existentes.

Todos esses fatores corroboram para maior estudo e possível utilização dessas tecnologias em meio acadêmico para que possivelmente sejam utilizadas a fim de melhorar a interação entre discentes e docentes à infraestrutura existente, no sentido de dar mais opções do que fazer com os recursos disponíveis.

1.2 Objetivos

Este trabalho tem como objetivo principal repensar a utilização de recursos de infraestrutura existentes do NTI afim de dar maior liberdade de opções, democratizando os recursos disponíveis do núcleo.

Para esse fim, propõe-se nesse trabalho uma arquitetura de nuvem, a partir dos recursos já existentes no CCET, que seja capaz de suprir as demandas atuais dos serviços fornecidos pelo suporte com características como isolamento entre projetos e criação de instâncias de computação para os usuários de forma nativa(com políticas de segurança por exemplo) e não de forma externa(com firewall por exemplo), como é realizada até certo nível atualmente. Na proposta apresentada são considerados requisitos como custo, segurança, disponibilidade e governança.

1.3 Organização do texto

O presente trabalho está estruturado em capítulos e, além desta introdução, possui a estrutura abaixo.

- Capítulo II: apresenta a história do surgimento do termo “Computação em Nuvem”, abordando seus principais serviços, origem e benefícios.

- Capítulo III: descreve a aplicação utilizada para construção das propostas, requisitos, descrição dos módulos.

- Capítulo IV: apresenta o processo de construção do caso de uso começando por uma introdução a estrutura organizacional do CCET e em seguida pela simulação de uso do OpenStack para um serviço específico do NTI utilizando computação em nuvem no laboratório.

- Capítulo V: apresenta as conclusões e considerações finais acerca da possibilidade da adoção e migração do ambiente atual, ou parte dele, para a nuvem.

2 CONCEITOS BÁSICOS

Esta seção apresenta um breve histórico do termo “Computação em nuvem”, abordando desde o surgimento das discussões das tecnologias até o momento da sua adoção como modelo de negócio pelas organizações.

2.1 Definições Iniciais de Nuvem

Apesar de ser um tema bastante difundido nos dias atuais, a origem do termo “computação em nuvem” vem de muito tempo e está atrelada ao surgimento da Internet. Foi na década de 1960 que o americano McCarthy trouxe ao mundo o conceito no qual duas ou mais pessoas pudessem usufruir de um mesmo computador (usar os mesmos recursos físicos) e que a computação pudesse ser ofertada como um serviço público à população.

Apesar de McCarthy ter sido pioneiro e de ter dado o pontapé inicial nas ideias a respeito da nuvem, somente em 1990 é que o termo começou a tomar forma. A ideia original foi inspirada na própria Internet “onde tudo estaria no ar”. Mais tarde em 2006, o conceito passou a ser também um modelo de negócio e marketing, se tornando realmente popular e sendo usado em diversos contextos.

2.2 História

Até o uso atual do termo “computação em nuvem” houve uma gradual evolução no que tange ao uso de recursos de infraestrutura principalmente armazenamento e processamento. Começando na era dos mainframes e aplicações de terminal, todo armazenamento de dados, valor de compra e manutenção desses sistemas por usuários eram muito custosos. A partir desses fatores, a solução foi aumentar a capacidade de armazenamento através de *Mass Storage* (Armazenamento Massivo) e prover acesso compartilhado a recursos para os usuários.

A partir de 1990, com o advento da Internet, os computadores foram se conectando e, consequentemente, criaram um pool de recursos que podiam ser compartilhados entre os usuários. Surgia neste momento a “*Grid Computing*” que, apesar de não ser computação em nuvem propriamente dita, solucionou parte dos problemas de compartilhamento de infraestrutura e recursos computacionais. O método consistia em dividir um único sistema em

diversas máquinas para atingir determinado objetivo. Durante um período, isso foi confundido com computação em nuvem.

Em 2006, Schmidt, CEO da Google, definiu o modelo ideal de negócio no qual os serviços seriam compartilhados pela Internet e teriam o seu valor de mercado [2]. A popularidade foi tanta que o conceito de computação em nuvem passou a ser utilizado em diversos contextos de marketing, gerando até mesmo uma certa confusão acerca do seu real significado.

Este trabalho adota o conceito proposto em [2] e que corresponde à definição mais adotada atualmente: nuvem é um *pool* (conjunto) de recursos alocados dinamicamente com garantia de escalabilidade, utilidade e flexibilidade. A utilização desses recursos pode ser paga (*pay-as-you-go* – pague o quanto utiliza) ou não.

2.3 Arquitetura da Computação em Nuvem

Antes de se contratar um serviço de computação em nuvem ou até mesmo implementar um para uso, é necessário que seja feito um estudo sobre o tipo de nuvem que será utilizado, uma vez que as necessidades e requisitos de negócio são inúmeros e diversos.

Na literatura, três tipos de modelos de computação em nuvem são abordados. Cada um desses tipos possui características específicas que auxiliam no momento de se selecionar o melhor modelo para o negócio.

1. **Nuvem Pública***:* os usuários acessam a nuvem através de um aplicativo-cliente; o acesso e o controle de custos são baseados na política *pay-per-use* (pague o quanto utiliza), ou seja, faz com que o usuário pague somente pelo o que ele realmente utiliza em determinado período. Isto reduz custos de operação de TI e custos de concessão de infraestrutura. Apesar da possibilidade de redução de custos de infraestrutura, este modelo esbarra em questões de segurança de dados (acesso e armazenamento), de rede e de privacidade. Exemplos conhecidos deste tipo de modelo são a AWS (Amazon Web Services) [8] e IBM Cloud [10].

2. **Nuvem Privada:** este modelo é uma forma de tornar mais confiável a implantação de estruturas na nuvem. A organização possui os recursos relacionados à infraestrutura, cabendo apenas definir o acesso a esses recursos e às aplicações lá abrigadas através da rede interna. Isto significa que a própria organização detentora do *datacenter* é a responsável pela administração dos recursos físicos. Por exemplo, o Openstack, que é um sistema de gerenciamento de nuvem cujo um dos objetivos é a manutenção e administração de nuvens privadas.

3. **Nuvem Híbrida:** é a combinação da Nuvem Pública com a Nuvem Privada. Ela consiste em ligar um recurso em Nuvem Privada a um ou mais recursos em Nuvem Pública. A vantagem é o fato de a organização poder armazenar dados críticos e de relevância ao negócio em nuvem privada e, ao mesmo tempo, ter a flexibilidade de aumentar ou diminuir o poder computacional através das nuvens públicas. Como exemplo, temos empresas como a IBM que dispõe de serviços para implementação de *clouds* híbridas implementadas em clientes por seus especialistas.

2.4 Tipos de Computação em Nuvem

A arquitetura em nuvem é normalmente dividida em quatro camadas (Hardware, Infraestrutura, Plataforma e Aplicação), onde uma camada fornece serviços, sob demanda, para a camada imediatamente acima. A Figura 1 mostra a disposição das camadas que são discutidas abaixo.

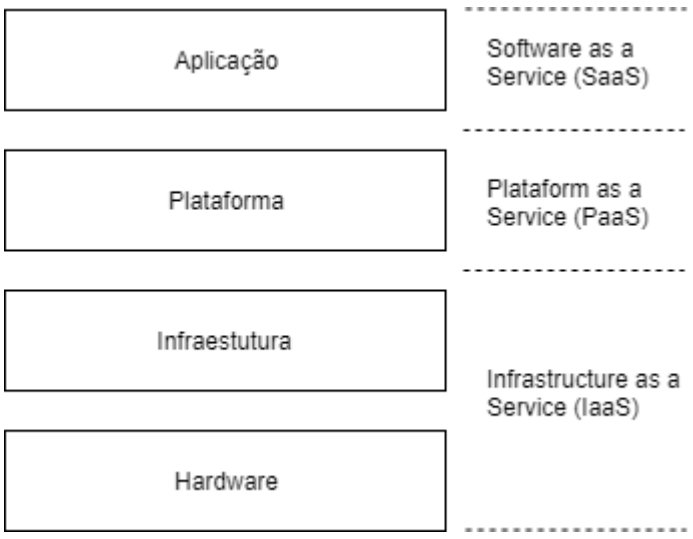


Figura 1: Arquitetura baseada em camadas da computação em nuvem

- **Camada de aplicação:** é a camada do topo da pilha arquitetural da computação em nuvem. É responsável pela entrega do serviço de *SaaS (Software as a Service).* Ela armazena as aplicações e fornece serviços específicos, como Multimídia, Web Services. Além disso, ela permite abstrair o acesso às aplicações através da Internet e de forma remota, eliminando assim a necessidade de instalação da aplicação localmente por parte de cada usuário. Como exemplo é possível citar o GSuite da google [31].

- **Camada de plataforma:** é a camada responsável pela entrega do serviço de *PaaS (Platform as a Service).* Ela consiste nos sistemas operacionais e aplicações de plataforma que permitem serviços como *storage*, banco de dados e outros recursos comuns ao desenvolvimento com interfaces de gerenciamento inteligentes. Como exemplo, pode ser citado o OpenShift da *RedHat*[30].

- **Camada de infraestrutura:** é a camada responsável por fornecer os serviços de *IaaS (Infrastructure as a Service).* É também chamada de camada de virtualização, pois possibilita a criação e controle dos *pools* de recursos de computação e de armazenamento. Ela possui todas características necessárias para interagir com o *hardware* abaixo dela através do uso de tecnologias de virtualização como *VMware* e *KVM.*

- **Camada de hardware:** é a camada responsável por organizar todos os recursos físicos que serão consumidos pelas camadas acima. É a estrutura que fica nos *datacenters.* Ela consiste em todos os servidores, *switches*, roteadores, energia e *storage*.

Além do proposto acima, o objetivo principal da computação em nuvem é atingir o maior nível de computabilidade possível usando recursos compartilhados e distribuídos a fim de evitar desperdícios. Neste trabalho será adotado, como estudo de caso da UNIRIO, um modelo de gerenciamento dos serviços do CCET como um IaaS (*Infrastructure as a Service*) na plataforma de código aberto OpenStack [7].

São características gerais desejáveis para a implementação de uma nuvem:

- Baixa complexidade e custo de implementação;
- Acesso a dados de infraestrutura como nós de computação, de rede e de administração via aplicativo de cliente;
- Uso eficiente de infraestrutura e barateamento de custos por conta do compartilhamento de recursos;
- Melhoria no processo de manutenção, pois a aplicação na nuvem não precisa estar presente no computador de cada usuário;
- Escalabilidade horizontal – possibilidade de adicionar mais nós computacionais a arquitetura conforme demanda, sem necessariamente adicionar um ou mais máquinas físicas; bem como a remoção automática quando ocorrer baixa demanda da aplicação.

2.5 Sistemas de Gerenciamento de Nuvem

Ainda dentro dos fundamentos da computação em nuvem, há um outro conceito importante relacionado a gerenciamento dos recursos, criação de regras, acesso de usuários e acesso aos serviços que estão diretamente relacionados à proposta abordada neste estudo. Isto é, o conceito de "Gerenciamento de Nuvem".

O gerenciamento de nuvem é um conceito relacionado a tudo que envolve a operação de uma nuvem. Isto é, os dados que são inseridos, as políticas de acesso dos usuários e as integrações com outros sistemas que podem consumir os recursos da nuvem. Estas aplicações podem ser externas à nuvem ou não [22].

Para o estudo de caso foram analisadas 3 ferramentas de gerenciamento de Nuvem:

- **Eucalyptus**: Ferramenta de código aberto compatível com a AWS (*Amazon Web Services*) para construção de nuvem privada e híbrida. Ela traz uma abordagem de *deploy* rápido e de baixa intrusão, de forma modular. Além disso, traz uma camada de rede capaz de separar diferentes tipos de tráfego, unindo em uma mesma LAN (*Local Area Network*) diferentes *clusters*, e traz APIs capazes de integrar com diferentes serviços terceiros, formando nuvem híbrida através de IaaS [23].

- **Open Nebula**: Ferramenta, também de código aberto, que tem o objetivo de gerenciar *data centers* virtualizados baseados em *KVM*, *LXD* ou *Vmware*. Ela pode ser usada para a criação de nuvens privadas, públicas ou híbridas. Sua funcionalidade e objetivos são parecidos com o do OpenStack, principalmente no que tange ao *design* de uma arquitetura baseada em IaaS. Entretanto, possui uma abordagem *all-in-one* (toda instalação em um mesmo ambiente), ou seja, os recursos que são modularizados no OpenStack, são colocados em uma arquitetura única no OpenNebula [27].

- **OpenStack:** Ferramenta de código aberto composta por módulos conectados através de APIs que gerenciam de forma distribuída os recursos da infraestrutura. Esta ferramenta foi escolhida para este trabalho por ser mais completa e capaz de abranger diferentes tipos de infraestrutura. Uma descrição mais detalhada do OpenStack é fornecida no próximo capítulo.

3 OPENSTACK

1.1 Introdução

O OpenStack é um sistema de gerenciamento de nuvem especializado em controlar grandes *pools* de computação, armazenamento e rede de um *datacenter* de forma modular e escalável. Todos os recursos são provisionados e gerenciados através de módulos e sua interação se dá através de mecanismos de autenticação.

O Openstack é um projeto de código aberto que teve inicialmente o objetivo de ser apenas uma *Application Program Interface (API)* compatível com os serviços da *Amazon Web Service (AWS),* onde as suas funções são expostas em *APIs REST*. Isto trouxe flexibilidade nas integrações com sistemas terceiros e nas automações e testes de software [6].

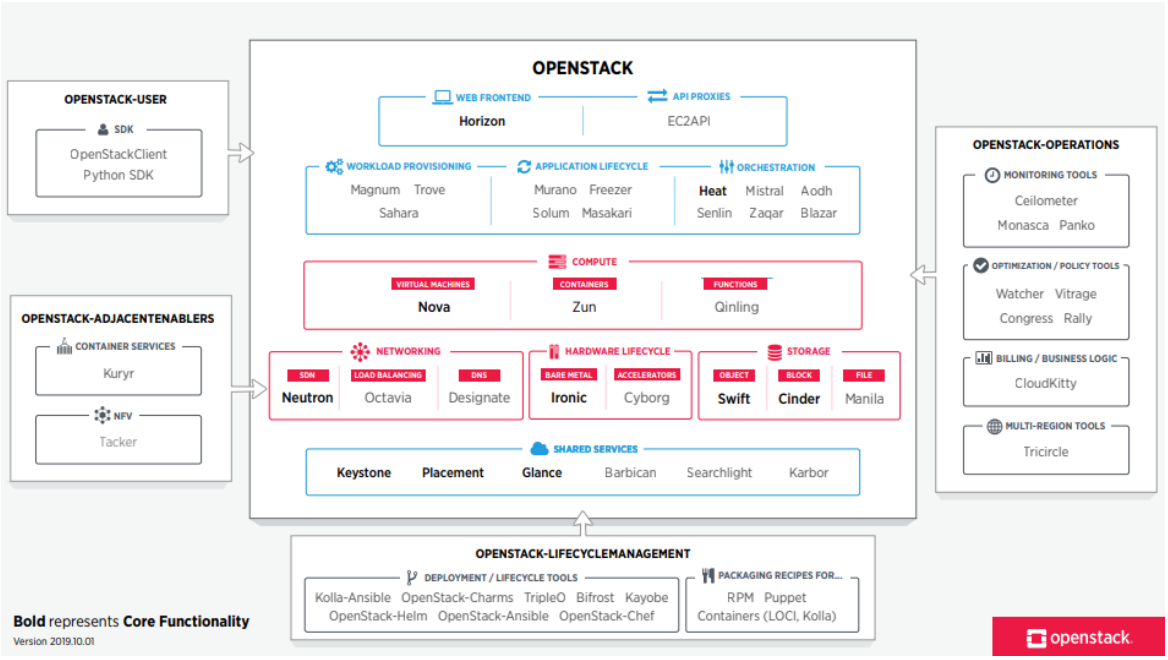
Além das flexibilidades apresentadas, o Openstack possui um projeto que recebe apoio da comunidade de usuários e de empresas como IBM e Red Hat. Esses apoiadores possuem um papel importante no desenvolvimento de *features* e de funções que interagem com partes específicas da Infraestrutura como o módulo de rede Nêutron [6] e na manutenção do projeto.

Para promover a integração da comunidade interessada no projeto, foi criada em 2012 a *OpenStack Foundation.* Ela centraliza o desenvolvimento e a distribuição do *software* para a comunidade, além de oferecer suporte aos usuários.

1.2 Componentes e Openstack como Nuvem Privada

O OpenStack, conforme foi relacionado nos tópicos anteriores, é um sistema modular. Apesar dele ter sido criado inicialmente como uma interface com a AWS, ele possui aspectos que o tornam capaz de criar uma nuvem privada. Isto é devido principalmente ao modelo de *tenant* ou projeto. Ou seja, toda instância virtual ou física gerenciada pelos serviços do OpenStack antes passa pelo módulo keystone de autenticação e autorização. Após a criação ou provisionamento destes recursos, o OpenStack cria um espaço privado com o objetivo de segregar o que está de fato sendo usado, principalmente recursos de rede, de armazenamento e de computação (somente para os projetos criados isoladamente). Isto é diferente de um *datacenter* que conta somente com virtualização [6], cenário encontrado atualmente no CCET.

Para que os recursos sejam criados, o OpenStack conta com estes módulos, onde cada um é responsável por gerenciar uma parte específica dos recursos de infraestrutura. Porém, em cada implementação, a utilização destes módulos depende do uso em que se está interessado, exceto quando se trata dos *core modules*. Estes módulos são fundamentais para o funcionamento do sistema, tendo que existir obrigatoriamente. Os módulos estão relacionados na Figura 2, sendo que os módulos principais são os descritos abaixo.



Fonte: Página oficial do OpenStack[1]

Figura 2: mapa de módulos OpenStack

- ***Web Frontend***:
  - **Horizon**: é uma interface web responsável por interagir diretamente com os serviços do OpenStack como Neutron (Rede), Swift (*Storage*), etc. Ele também fornece recursos capazes de gerenciar todo o cluster OpenStack de forma simplificada, sobretudo através de *dashboards*[32].

- **Computação**:
  - **Nova**: é um dos componentes mais importantes do OpenStack, ele é responsável diretamente pelo provisionamento de máquinas virtuais chamadas de instâncias, *containers* de aplicação ou recursos físicos. Para o funcionamento do

1 Disponível em:<https://www.openstack.org/software/>. Acesso em: 29 de out. 2019

provisionamento, o OpenStack interage diretamente com o módulo de imagem (*Glance*) e segue um *template* de *hardware* específico, chamado de *flavor (*depende do caso de uso). O *Flavor* é importante, pois descreve as características a nível de *hardware* que uma instância vai possuir como, por exemplo, quantidade de memória, de disco e de processamento [6].

- **Storage:**
  - **Swift (*Object Storage*):** é o módulo utilizado para armazenamento de dados de forma escalável e redundante em um *cluster* de servidores a nível de *Software.* Isto significa que, quando for necessário um maior armazenamento basta apenas adicionar mais um nó no *cluster*. O *object storage* é ideal para o armazenamento, de objetos, ou seja, desde vídeos e músicas à imagens de criação de instância do próprio openstack, SSH Keys, entre outros objetos[33].

  - **Cinder (*Block Storage*):** é o módulo responsável pelo armazenamento tradicional. O módulo pode criar volumes que podem ser adicionados ou removidos em uma ou mais instâncias. Isto significa que o volume é mantido independentemente das instâncias [6].

- **Rede:**
  - **Neutron:** é responsável por fornecer os serviços de rede. Ele provê as funcionalidades de criação de portas, subrede, redes e roteadores além de endereçamento. Outros componentes como *Firewalls* e *VPN* também podem ser implementados[6].

- ***Shared Services*:**
  - **Keystone:** é responsável pelo serviço de Identidade e Autorização, sobretudo com relação aos *tenants* (projetos). O keystone funciona com um esquema de "tokenização". Ou seja, para cada operação (ex: instanciação, criação de rede, criação de usuário), é gerado um token de autenticação que será usado para permitir determinada operação no Openstack [27].

  - **Glance**: é responsável por prover todo os serviços de imagem e de definições de metadados. Esse módulo permite descobrir, gravar e consultar uma imagem

específica de uma instância [26].

Apesar de não terem sido detalhados, o uso dos outros módulos depende do contexto em que se está implantando o OpenStack, podendo ser adicionados mais recursos ou funcionalidades específicas em uma nuvem privada.

1.3 Distribuições

Conforme apresentado na seção 3.1, o Openstack é um projeto que recebe inúmeros incentivos, seja da comunidade de usuários ou de empresas apoiadoras. Antes de iniciar um projeto com o OpenStack, é necessário entender as diversas possibilidades de aquisição da plataforma que em sua maioria utiliza distribuições do Linux como sistema operacional. Atualmente são ofertadas as seguintes possibilidades:

- ***Community**:* Este é o caso mais comum, onde a organização busca a versão baseada em uma determinada distribuição do Linux. Após a escolha, a organização consegue instalar e configurar o OpenStack com a documentação disponível. A vantagem é a disponibilidade e custo zero. Por exemplo, existe a distribuição de comunidade RDO (*RPM distributed Openstack*), que atende de forma completa o *deploy* (utilização) do sistema [6].

- ***Enterprise:*** Além das versões da comunidade, há aquelas onde uma organização específica realiza o desenvolvimento de uma plataforma e a torna um produto comercial. A principal vantagem sobre as versões de comunidade é o processo de garantia de qualidade de software. Além disso, a organização possui uma grande base de conhecimento de resoluções de problemas devido aos inúmeros testes realizados antes do lançamento e pode fornecer correções rápidas dos bugs [6]. Outro ponto importante é o atendimento com um SLA (*Service Level Agreement*) personalizado para o comprador do *software*, garantindo assim a manutenção do ambiente.

4 ESTUDO DE CASO

Esta seção apresenta o estudo de caso do CCET. Inicialmente são discutidos a estrutura administrativa e o modelo de gestão do CCET. Em seguida, são abordados a infraestrutura e os serviços disponibilizados aos docentes, discentes e técnicos administrativos com o uso da tecnologia de virtualização de servidores. A partir desse cenário são apresentadas duas propostas de migração desses serviços para um ambiente de computação em nuvem com a aplicação OpenStack.

4.1 Modelo de Gestão

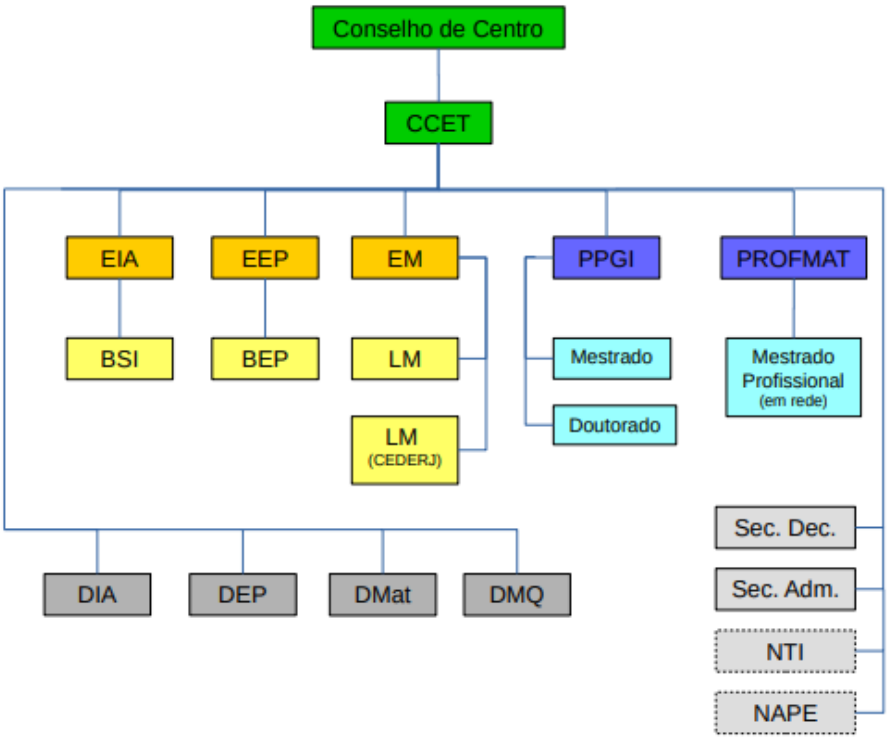
O Centro de Ciências Exatas e Tecnologia (CCET) é um centro acadêmico responsável pelas escolas, cursos de graduação e pós-graduação na área de tecnologia. Fazem parte do CCET as Escolas de Informática Aplicada (EIA), Escola de Matemática (EM) e a Escola de Engenharia de Produção (EEP). Essas escolas são responsáveis, respectivamente, pelos cursos de Bacharelado em Sistemas de Informação (BSI), Licenciatura em Matemática (LM) e Bacharelado em Engenharia de Produção (BEP). Além desses cursos, há um curso de Licenciatura em Matemática a distância através do consórcio CEDERJ e três cursos de Pós-Graduação.

Dentro dos programas de pós-graduação ofertados, há o Programa de Pós-Graduação em Informática (PPGI), cujas modalidades são o Mestrado Acadêmico e Doutorado Acadêmico em Informática, ambos com pesquisas focadas na área de Sistemas de Informação, subdividindo-se em 3 linhas: Distribuição de Redes (DR), Representação do Conhecimento e Raciocínio (RCR) e Sistemas de Apoio a Negócios (SAD). Outro programa de Pós-Graduação é o de Mestrado Profissional em Matemática (PROFMAT) ligado à EM que visa aperfeiçoar a formação dos professores de matemática.

Além das Escolas, Programas de Pós-Graduação e secretarias, o CCET em nível de gestão se subdivide em Decania, Conselho de Centro, Secretaria Administrativa, Núcleo de Assuntos Pedagógicos e Educacionais (NAPE) e Núcleo de Tecnologia da Informação (NTI). Apesar da importância em termos de definição no que tange às funções de cada subdivisão do CCET, neste trabalho vamos focar no Núcleo de Tecnologia da Informação (NTI), cujo objetivo é cuidar de questões ligadas a projeto, implantação e suporte de serviços de TI atendendo a

demandas do próprio CCET e também do Instituto de Biociências (IBIO) [5].

A Figura 3 mostra a estrutura organizacional do CCET discutida acima.



Fonte: Relatório Gestão do CCET Exercício 2017[2]

Figura 3: Estrutura organizacional do CCET

4.2 NTI (Núcleo de Tecnologia da Informação)

Para o estudo de caso, foi levado em consideração a importância do NTI como provedor de serviços e como figura de inovação para o CCET, sobretudo no que tange ao fornecimento de recursos tecnológicos como *DNS*, e-mail institucional, *moodle* e infraestrutura para pesquisa.

Os serviços fornecidos pelo NTI para a comunidade acadêmica usam a tecnologia de virtualização VMware VSphere. Isto significa que cada serviço é disponibilizado por meio de uma ou mais máquinas virtuais onde os recursos são pré-alocados e obedecem um padrão pré-definido pelo suporte do NTI. Isto faz com que a arquitetura atual tenha pouca elasticidade a nível arquitetural, uma vez que a cada melhoria deve ser adicionada outra VM diferente da primeira, além de dar pouca liberdade e flexibilidade ao usuário para gerenciar recursos.

2 Disponível em: < http://wwwp.uniriotec.br/ccet/wp-content/uploads/sites/39/2019/05/Relat%C3%B3rio-de-Gest%C3%A3o-do-CCET-exerc%C3%ADcio-2017.pdf" >. Acesso em: Setembro, 2019

4.3 Fatores para adoção do Modelo

A adoção de uma arquitetura baseada em nuvem significa modificar a forma como os recursos são disponibilizados para os usuários. A arquitetura passaria a contar com características como proteção aos serviços (redundância, balanceamento de cargas, zonas de disponibilidade, etc.) e auto escalonamento (aumento ou diminuição do poder computacional dos serviços).

Note que há uma mudança de paradigma com a mudança da tecnologia de virtualização para a tecnologia de computação em nuvem. O modelo atual aloca máquinas prontamente virtualizadas sem margem para alterações ou customizações (é preciso criar novas máquinas para mudar as alocações dos recursos). O modelo proposto de computação em nuvem aloca recursos virtualizados para os projetos. Isto significa que é o administrador do projeto quem define como os recursos serão utilizados.

Como exemplo, imagine um professor que está envolvido em duas diferentes pesquisas. Para a primeira pesquisa é necessário alocar 10 Gbytes de disco, enquanto para a segunda pesquisa são necessários 100 Gbytes. No ambiente de máquinas virtuais, o suporte teria que criar duas máquinas virtuais para atender às duas pesquisas. Caso o pesquisador precisasse transferir espaço em disco de um projeto para o outro, seria necessário recriar as duas máquinas virtuais. Na tecnologia em nuvem, o suporte alocaria os 110 Gbytes e o pesquisador teria liberdade de distribuir o espaço em disco da forma que desejasse.

É importante notar que existem indicadores que devem ser levados em consideração na hora de decidir pela tecnologia de nuvem. Para este projeto, foram analisados os seguintes indicadores nas próximas seções: custo, performance, segurança, disponibilidade e *hipervisor*.

4.4 Análise de custo

No geral, o custo é o primeiro indicador analisado quando se trata de arquiteturas em nuvem, pois ele interfere diretamente no tipo de contratação e no tipo de Nuvem que se está planejando implantar. Além disso, quanto maior a complexidade da Nuvem, maior será o valor que deve ser investido, pois ela deverá possuir maior poder computacional para atender as regras do negócio da organização [24].

Para o projeto em questão, não foram feitos investimentos com relação a arquitetura, pois além de se tratar de uma proposta de modelo (não é uma implantação comercial), foi

selecionado o modelo de código aberto OpenStack *Community*. Entretanto, é importante ressaltar que uma vez que o modelo seja implantado, pode ocorrer dele ser expandido para toda a Universidade. Neste caso, deve-se considerar para fim de compra [25]:

- Poder computacional
- Rede
- Replicação e Redundância
- Armazenamento
- Backup

4.5 Análise de Performance

A performance é um dos fatores mais importantes para esse estudo uma vez que a arquitetura atual possui múltiplas aplicações que realizam diferentes funções todo o tempo.

Por se tratar de uma arquitetura de administração fechada, não foram expostos mais detalhes sobre a carga das aplicações no ambiente, como por exemplo *throughput,* consumo de largura de banda entre outras métricas. Por isso foram considerados apenas detalhes físicos gerais da arquitetura, ou seja, detalhes inerentes aos servidores.

Dentro os dados físicos dos servidores, como quantidade de memória RAM, foram calculadas a quantidade de nós computacionais e nós controladores que cada servidor suportaria sem tornar o OpenStack inconsistente.

Além disso, houve uma preocupação com a distribuição dos recursos disponíveis para as aplicações já existentes como portais, e-mail institucional, DNS, etc. E por fim, foram analisadas as questões relacionadas ao armazenamento e a configuração da rede, sendo o primeiro responsável por garantir o armazenamento dos dados das aplicações e do sistema e o segundo por garantir a redundância, acesso e segurança ao sistema e a seus recursos.

4.6 Análise de Segurança

Neste tópico foram analisadas questões relacionadas à segurança do ambiente como acesso às aplicações do NTI, acesso e configuração dos projetos e acesso aos módulos do OpenStack.

Devido aos diferentes serviços existentes no NTI, foi necessário criar diferentes projetos com o objetivo de segmentar os acessos às aplicações. Por exemplo, para o serviço de "hospedagem de servidores e páginas" para as escolas do CCET, um projeto "Escolas CCET" foi criado com perfis específicos representando alunos e professores para acesso às instâncias.

4.7 Análise de Disponibilidade

O objetivo é a adoção de uma arquitetura de alta disponibilidade. Isto significa que o hardware e o software em conjunto contribuem para que o sistema esteja sempre *online* e o usuário mantenha a experiência de uso das aplicações mesmo em condições de falha sistêmica ou física.

Na arquitetura atual do NTI, estão instalados 3 servidores físicos com hipervisor *Vmware VSphere*, 48GB de Memória RAM e 300 GB de Disco rígido, armazenados em Storages redundantes. A proposta é utilizar dois destes três servidores com função principal de serem os nós computacionais da arquitetura. Já o terceiro servidor será a controladora do ambiente, ou seja, ele será o responsável por gerir as requisições de criação no ambiente e em caso de problemas de software, físicos, rede ou até mesmo com algum nó computacional, o módulo irá redirecionar as requisições para o outro servidor redundante. Para que a controladora não seja um ponto único de falha, a aquisição de mais um servidor para controladora seria necessário de forma a se ter a redundância dos mesmos e continuar com a proposta de diminuição dos pontos de falha. Entretanto, como a proposta da Universidade não é no momento investir em hardware, os servidores de controladora serão virtualizados no terceiro servidor.

Os acessos às aplicações embarcadas no ambiente serão realizados através de um IP Virtual único, configurado em um balanceador de carga, no caso o HA Proxy [14], também virtualizado, que terá o papel de gerenciar o tráfego e encaminhar as requisições para um dos

dois ambientes computacionais caracterizando, portanto, a arquitetura Ativo/Ativo [12] proposta.

A nível de hardware, o ambiente deve contar com diferentes configurações para que seja efetiva a alta disponibilidade proposta. Além de contar com a redundância do *storage* já existente na arquitetura atual, é necessário contar também com uma redundância a nível de *switch*, pois faz com que os problemas de redirecionamento do tráfego sejam diminuídos em caso de falha em algum elemento que esteja nestas redes.

Este tipo de redundância é importante pois o usuário não sentirá nenhum *downtime* em caso de algum evento na rede, mantendo uma boa experiência de uso. Outro fator importante também é que devem ser configuradas interfaces "*bond*", que são interfaces físicas de rede independentes nos servidores, pois caso uma interface de rede no modo ativo/ativo tenha uma falha, a segunda interface *bond* assume.

4.8 Análise de Hipervisor

Outro fator indispensável nesta análise é a escolha do hipervisor, que será o responsável por ser o virtualizador do ambiente e estar diretamente relacionado ao módulo *nova* do OpenStack.

De forma nativa, o OpenStack vem com o hipervisor KVM no pacote, atendendo a maioria dos *deployments*. Por ser uma tecnologia também *open-source,* a migração do virtualizador VSphere seria uma opção e um fator de economia, uma vez que não haveria a necessidade de se investir mais principalmente em licenciamento, mas por questões de reutilização da infraestrutura disponível, foi concluída a utilização e a integração do Hipervisor existente com a *API* do módulo *nova*.

4.9 Proposta da Arquitetura

Dentro deste tópico será apresentada a montagem da arquitetura proposta para o CCET, levando em consideração os requisitos de *design* discutidos anteriormente.

Conforme os requisitos de *design*, o ambiente OpenStack será organizado da seguinte maneira:

- **Nó de controladora:** são os nós responsáveis pelos serviços de *API,* interface web (módulo Horizon) e por controlar os nós computacionais, é denominado de *Control Plane.* Conforme explicado acima, serão 2 máquinas virtuais hospedadas no servidor de controladora. O *control plane* em um ambiente de nuvem de alta

disponibilidade geralmente gerencia as requisições de autenticação, *mensageria* entre os sistemas e outros serviços como gerência de requisições para o banco de dados (persiste dados de status da nuvem) e serviços de gerencia de imagens.

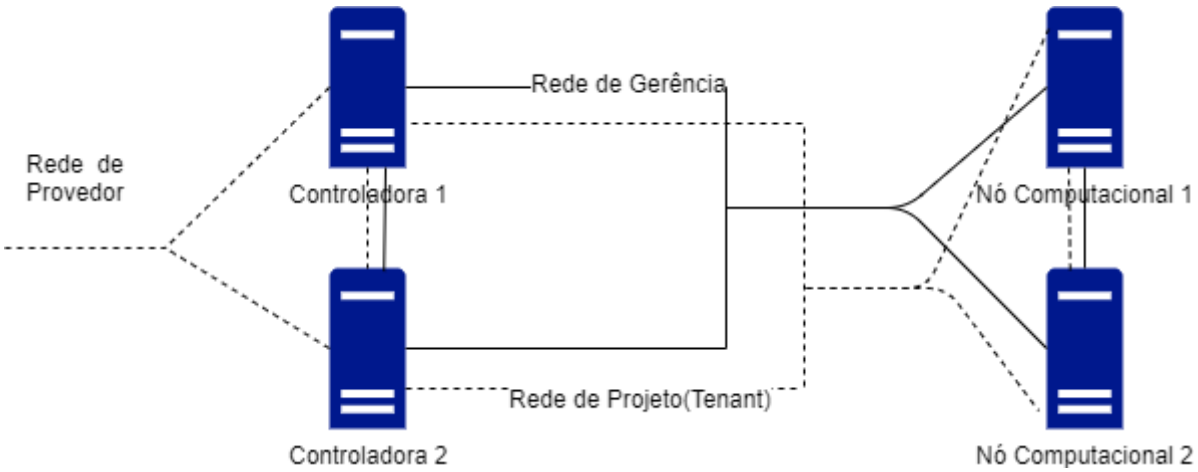
- **Nós computacionais:** são os nós responsáveis pelo processamento e por criar instâncias na nuvem. Nele estará presente a figura do *hipervisor*, no caso, o VSphere.

- **Rede de Gerência:** Rede responsável por manter a comunicação, privada, entre os nós do ambiente. Ela também gerencia o tráfego que chegam aos projetos (*tenants)* e ao *storage* [15].

- **Rede de provedor:** Responsável por fornecer os acessos às instâncias de um projeto específico, através de portas públicas configuradas no segmento de rede da controladora [15].

- **Rede de projetos (*Tenant*):** Rede que vai gerenciar o tráfego dos projetos no ambiente, ou seja, vai fazer o mapeamento da rede física com a rede virtual específica criada para um projeto [16].

Sendo assim, a configuração lógica, considerando a organização acima será como mostrada na Figura 4 que apresenta principalmente o esquema da rede.



Fonte: Livro OpenStack for Architects ed. 1, 2017.

Figura 4: Disposição inicial dos servidores e da rede na proposta de arquitetura CCET

Além disso, a distribuição dos recursos de cada servidor será conforme a Tabela 1 que reflete as configurações de rede e de servidor descritas anteriormente [15].

Nome do Servidor	CPU	Memoria	Disco	Interface
VM-Controladora 1	8	16GB	200GB	2x. Giga
VM-Controladora 2	8	16GB	200GB	2x. Giga
Nó computacional 1	8	48GB	300GB	2x. Giga
Nó computacional 2	8	48GB	300GB	2x. Giga

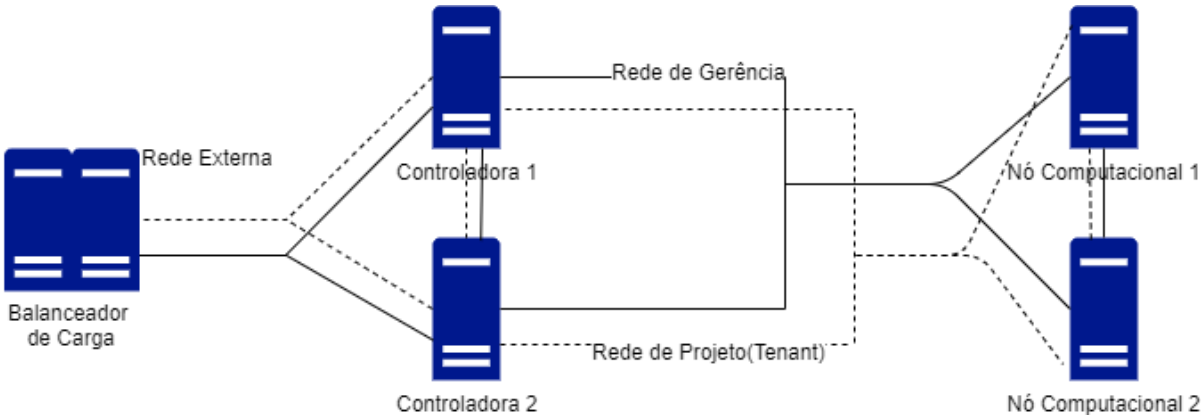
Tabela 1: Requisitos físicos dos servidores CCET.

Apesar do modelo acima, vale ressaltar a necessidade de replanejamento constante da arquitetura, uma vez que um dos benefícios da alta disponibilidade é a possibilidade de escalonamento horizontal, ou seja, a capacidade de se adicionar ou diminuir um recurso do ambiente sem comprometer o funcionamento das aplicações hospedadas via adicionando novos nós ou retirando eles, ao invés de aumentar ou diminuir recursos de hardware de nós existentes.

Sendo assim, apesar da arquitetura acima estar atendendo à demanda atual, é importante que sempre haja avaliações cada vez que a estrutura aumenta ou é mais requisitada. A proposta atual então visa atender os seguintes padrões:

- As chamadas aos serviços devem continuar funcionando mesmo existindo um problema de hardware na controladora.
- O poder computacional pode ser aumentado ou diminuído horizontalmente ou verticalmente
- Os alunos e professores, devem conseguir utilizar os serviços e portais mesmo com uma falha de hardware na controladora.
- Por conta de as controladoras serem virtualizadas, os mesmos só permitem escalonamento vertical, ou seja, melhorar o hardware disponível [15].

Após estas definições, com a adição do balanceador de carga, a arquitetura de alta disponibilidade, considerando os fatores de escalonamento acima, é apresentada na Figura 5.



Fonte: Livro OpenStack for Architects, ed.1, 2017.

Figura 5: Disposição definitiva dos servidores e da rede na proposta de arquitetura CCET

A relação dos servidores do segundo modelo fica conforme mostra a Tabela 2.

Nome do Servidor	**CPU**	**Memoria**	**Disco**	**Interface**
VM-Controladora 1	3	16GB	60GB	2x. Giga
VM-Controladora 2	3	16GB	60GB	2x. Giga
Nó computacional 1	8	48GB	300GB	2x. Giga
Nó computacional 2	8	48GB	300GB	2x. Giga
VM - HAProxy 1 (Balanceador de Carga)	2	8GB	30GB	2x. Giga

Tabela 2: Relação de servidores na arquitetura de alta disponibilidade.

Com a arquitetura acima, o CCET deverá ter a resiliência necessária para atender com qualidade as demandas operacionais e de uso dos usuários do Centro. Uma vez que ela conta com redundância física e elasticidade, visando reduzir a maioria dos pontos de falha, existentes hoje, a arquitetura ativo/ativo traz uma nova forma de administração e entrega de serviços.

A partir da criação do modelo são observadas as seguintes vantagens relacionadas a administração do ambiente:

- Investimento em rotinas de automação de *deploy* de servidores e serviços.
- Possibilidade de automação das rotinas de atualização das aplicações sem *downtime.*
- Diminuição no custo de manutenção.
- Maior ROI (Retorno de Investimento)
- Maior flexibilidade e possibilidade de crescimento

4.10 Organização da Arquitetura Com Virtualização

Apesar da arquitetura acima já atender os principais requisitos discutidos neste trabalho, pode-se adotar uma arquitetura com os componentes virtualizados. Isto é, a criação de uma arquitetura totalmente baseada em máquinas virtuais.

Os três servidores disponíveis serviriam de insumo para máquinas virtuais que fariam as funções de controladora e nó computacional. A vantagem principal é que o *control plane* seria melhor distribuído entre instâncias, permitindo uma resiliência maior que a da arquitetura com máquinas virtuais em apenas em um servidor físico.

Outro ponto seria a facilidade de se adicionar nós computacionais ao ambiente já existente, ou seja, pode-se iniciar a arquitetura com 2 nós computacionais e à medida que o ambiente necessitar de mais poder computacional, adiciona-se mais máquinas virtuais (escalonamento horizontal).

Sendo assim, esta arquitetura seria organizada conforme a topologia de um servidor com as características dos nós virtualizados.

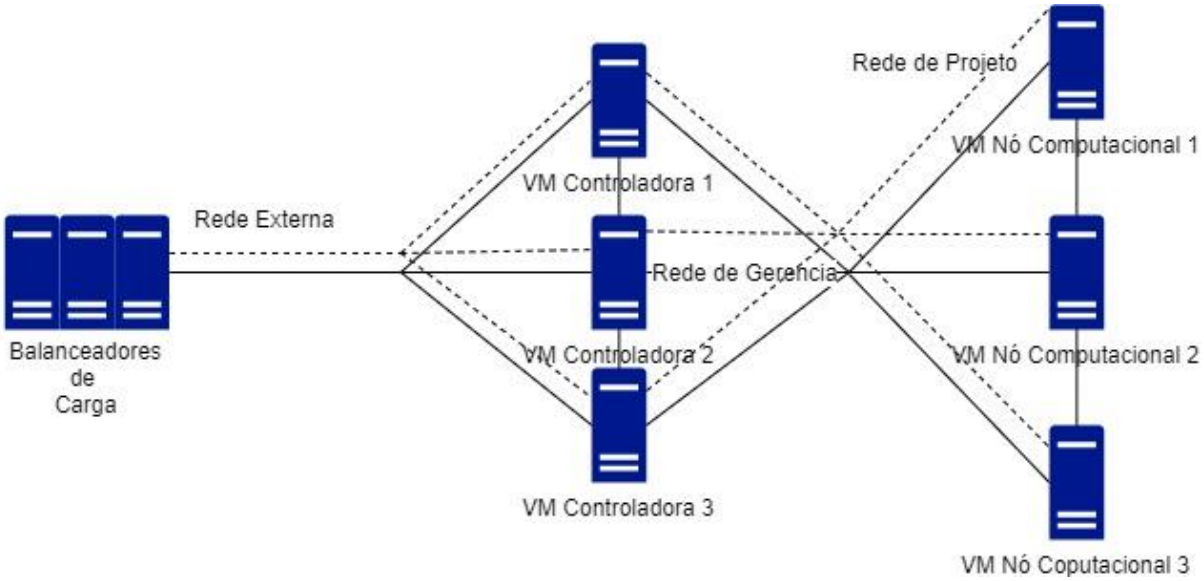


Figura 6: Disposição definitiva dos servidores e da rede em proposta de arquitetura CCET – Virtualizado

Com relação às características de cada máquina virtual que hospeda os nós computacionais e controladores virtuais, um exemplo inicial se dá conforme tabela 3 abaixo.

Nome do Servidor	**vCPUs**	**Memoria**	**Disco**	**Interface Do Servidor Físico**	**Hospedado em:**
VM-Controladora 1	1	4GB	30GB	2x. Giga	Servidor 1
VM-Controladora 2	1	4GB	30GB	2x. Giga	Servidor 2
VM-Controladora 3	1	4GB	30GB	2x. Giga	Servidor 3
VM - Nó computacional 1	6	42GB	240GB	2x. Giga	Servidor 1
VM - Nó computacional 2	6	42GB	240GB	2x. Giga	Servidor 2
VM - Nó computacional 3	6	42GB	240GB	2x. Giga	Servidor 3
VM - HAProxy 1 (Balanceador de Carga)	1	2GB	30GB	2x. Giga	Servidor 1

VM - HAProxy 2 (Balanceador de Carga)	1	2GB	30GB	2x. Giga	Servidor 2
VM - HAProxy 3 (Balanceador de Carga)	1	2GB	30GB	2x. Giga	Servidor 3

Tabela 3: Relação de servidores na arquitetura de alta disponibilidade virtualizada.

A partir da configuração acima, a alta disponibilidade continua sendo dada a nível de OpenStack, ou seja, as rotinas de replicação, redirecionamento de tráfego e detecção de anomalias de *software* e rede continuam. Porém, em comparação a primeira arquitetura, a configuração do hipervisor será realizada através do KVM para as VMs que representam os nós computacionais, não sendo realizada qualquer integração com VSphere instalado no ambiente existente, sendo este apenas responsável pela gerência de recursos da nuvem privada.

4.11 Simulação do Estudo de Caso em Laboratório

Visando verificar o funcionamento do OpenStack e dos conceitos de computação em nuvem apresentados, foram realizados testes em laboratório com um *deploy* de uma arquitetura *all-in-one do OpenStack*, ou seja, com recursos de controladora e nó computacional em um mesmo servidor.

O servidor utilizado para a simulação foi virtualizado utilizando o *software* Virtual Box e a distribuição do OpenStack *Stein*[34]. Os requisitos desta máquina virtual estão definidos na Tabela 4.

Nome do Servidor	**CPU**	**Memoria**	**Disco**	**SO**
VM-Teste 1	5	8GB	300GB	CentOS 7

Tabela 4: Requisitos da máquina virtual do caso de uso

4.12 Configuração do Laboratório

Antes de iniciar o processo de instalação do OpenStack é necessário ter atenção no processo de configuração de rede da instância de teste durante o processo de instanciação da máquina virtual. Para as configurações de rede foi realizada a configuração da interface de modo a permitir qualquer conexão a ela a partir da rede local ao servidor remoto e a fixação do IP. A Figura 7 abaixo apresenta o resultado da configuração da interface da máquina virtual.

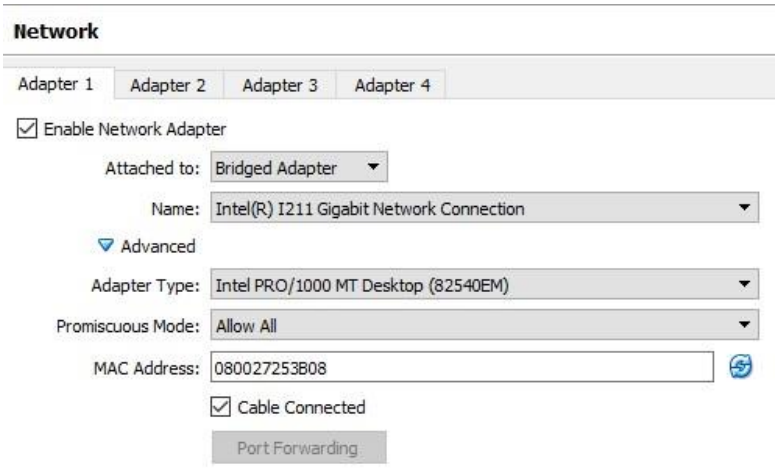


Figura 7: Configuração de rede da máquina virtual do laboratório

Quanto a configuração do sistema operacional, não foi realizada nenhuma especificação a nível de partição, porém a nível de configuração de rede e segurança foram realizadas duas:

- Fixação de IP da máquina virtual e criação de rota estática
- Desativação das configurações *default* de segurança do sistema operacional.

As configurações são mostradas nas Figuras 8 e 9.

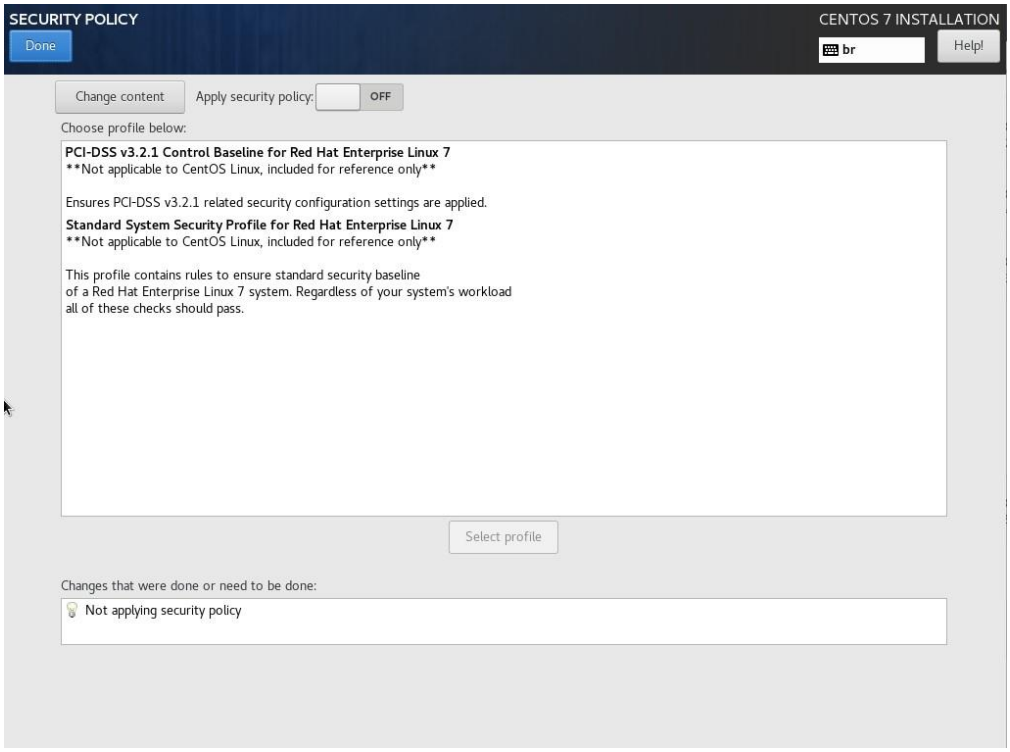


Figura 8: Desativação das políticas default de segurança do S.O (iptables)

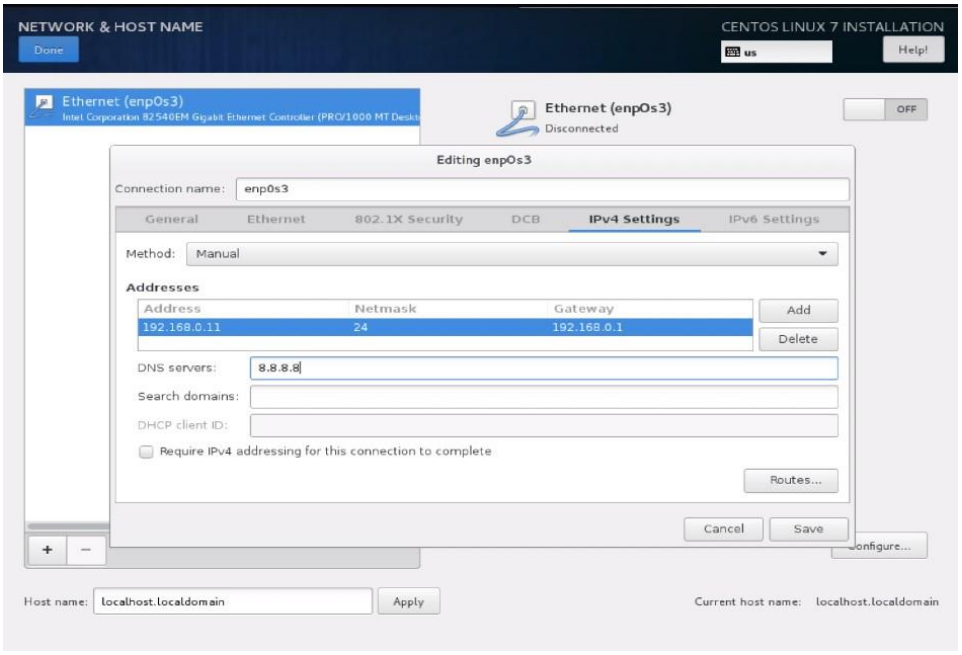


Figura 9: Fixação de IP do laboratório e rota estática

Após processo de instalação do sistema operacional, foram realizadas operações de preparação do ambiente para instalação da aplicação. Dentre elas, desativação do *firewall* do servidor, serviço *NetworkManager*(gerencia os recursos de rede do Linux) *e Selinux*(gerencia as regras de acesso do servidor)*:*

```
#Checa o status do serviço firewalld. Se estiver habilitado pare e desabilite
    systemctl status firewalld
    systemctl stop firewalld
    systemctl disable firewalld

# Checa o status do serviço NetworkManager e desativa caso esteja ativado

    systemctl status NetworkManager
    systemctl stop NetworkManager
    systemctl disable NetworkManager

# Habilitar o serviço de rede e iniciar o serviço
    systemctl enable network
    systemctl start network
```

```
# Desabilite o serviço do selinux
vi /etc/selinux/config
SELINUX=disabled
```

Após as configurações acima, o processo de *deploy* do ambiente inicia a partir do *download* do repositório RDO OpenStack, seguindo o comando abaixo:

```
yum install -y https://rdoproject.org/repos/rdo-release.rpm
```

Com o repositório instalado o *deploy* do OpenStack seguiu conforme configuração abaixo iniciando pela instalação do executável correspondente à distribuição escolhida, no caso o ***OpenStack-Stein***, bem como o instalador *Packstack*:

```
sudo yum-config-manager --enable openstack-stein
#Realiza a atualização dos pacotes
sudo yum update -y
#instala o intalador Packstack
sudo yum install -y openstack-packstack
```

Após a instalação do *Packstack,* foi executado o comando abaixo para de fato realizar o *deploy* da aplicação, bem como foi realizado o processo de configuração da rede utilizando OVS (*Open V-Switch*), cujo objetivo é implementar uma rede OpenStack utilizando ML2(*Modular Layer-2*). A implantação deste tipo de rede consiste em permitir que usuários não administradores gerenciem suas redes virtuais dentro de um projeto além de delimitar as interfaces que terão o acesso a rede externa, como a internet e rede de gerenciamento.

Como este é um caso de laboratório e o servidor de teste contemplou apenas a configuração de uma interface de gerenciamento e externa. Porém, em um ambiente de produção deve-se realizar uma avaliação quanto ao tipo de configuração da rede se será vxlan, vlan, ou flat, pois influencia na forma com o que agente do módulo *nêutron* vai interagir com os componentes e até mesmo na criação de redes dentro de um projeto.

Sendo assim, para o laboratório foram realizadas as seguintes configurações:

```
vi /etc/sysconfig/network-scripts/ifcfg-enp0s3
TYPE=OVSPort
NAME=enp0s3
DEVICE=enp0s3
DEVICETYPE=ovs
OVS_BRIDGE=br-ex
ONBOOT=yes
```

Abaixo a configuração da *Linux-Bridge* para a interface acima:

```
vi /etc/sysconfig/network-scripts/ifcfg-br-ex
DEVICE=br-ex
DEVICETYPE=ovs
TYPE=OVSBridge
BOOTPROTO=static
IPADDR=<your_IP>
NETMASK=<máscara de rede>
GATEWAY=<gateway de rede>
IPV4_FAILURE_FATAL=no
IPV6INIT=no
DNS1=8.8.8.8
ONBOOT=yes
```

Após serem realizadas as configurações físicas, foram executados os comandos abaixo para instalação definitiva do OpenStack utilizando o Packstack, sendo um dos parâmetros a rede configurada acima:

```
packstack --allinone --provision-demo=n --os-neutron-ovs-bridge-mappings=extnet:br-ex --os-neutron-ml2-mechanism-drivers=openvswitch --os-neutron-l2-agent=openvswitch --os-neutron-ovs-bridge-interfaces=br-ex:enp0s3 --os-neutron-ml2-type-drivers=vxlan,flat --os-neutron-ml2-tenant-network-types=vxlan
```

Assim que finalizadas as instalações foi necessário se autenticar como administrador do OpenStack utilizando os arquivos de autenticação gerados durante a instalação:

```
source keystonerc_admin
```

Depois foi necessário criar uma rede utilizando o nêutron para que as instancias que fossem criadas à posteriori se comunicassem com o ambiente externo, ou seja, criar a rede de provedor. Neste caso, o parâmetro será um *range* de IPs válidos, específicos da rede em que o OpenStack está sendo instalado:

```
neutron net-create external_network --provider:network_type flat --
  provider:physical_network extnet --router:external

neutron subnet-create --name public_subnet --enable_dhcp=False --allocation-pool
  start=192.168.0.200,end=192.168.0.240 --gateway=192.168.0.1 external_network
  192.168.0.0/24
```

Após estes procedimentos, o acesso a interface web do OpenStack (*Horizon*) já pode ser estabelecido.

4.13 Desenvolvimento do Caso NTI

Com o acesso estabelecido ao ambiente do OpenStack, os próximos passos da pesquisa foram o de configurar um ambiente similar ao que seria praticado no dia-a-dia do NTI e verificar como os recursos do OpenStack fornecem insumos para que haja um processo e governança melhores. Para que fossem apresentadas questões, foram criados 3 projetos (*tenants*) com o objetivo de representarem projetos de pesquisa das outras escolas pertencentes ao CCET, recursos de segurança, isolamento entre projetos, comunicação e rede.

O primeiro passo do estudo de caso foi simular um fluxo de requisição de um servidor vinda de uma escola do CCET, passando pelo processo de análise e entrega. No caso, a necessidade é atender a um projeto de pesquisa que consiste em criar um portal *web* para um programa de ensino.

Ao receber a demanda através do “Sistema de Atendimento”, o NTI analisa a requisição do usuário, de forma a entender quais são os requisitos da demanda a fim de tomar a melhor

decisão e qual solução deverá ser aplicada. Os requisitos aplicados para esta demanda são apresentados na Tabela 5.

Usuário	**Descrição**	**Regra de Negócio**	**Objetivo**	**Pós-Requisito**
Professor responsável pelo projeto e pesquisadores	Servidor para desenvolvimento de pesquisa com suporte à aplicação *web.*	• Apenas os usuários envolvidos no projeto devem ter acesso ao ambiente de desenvolvimento • O portal final deve ser acessado pelos alunos do CCET • O professor deve ser o usuário administrador do projeto, podendo criar e modificar o ambiente.	Criação de um portal WEB para divulgação de programa de ensino.	• O Portal deve estar acessível apenas pela comunidade acadêmica da UNIRIO.

Tabela 5: Representação dos requisitos de demanda de servidor para escola CCET

Analisando a demanda, foi entendido que a demanda consiste em realizar o provisionamento de uma instância com um servidor *web* instalado, onde o professor seria o gerente do projeto e os alunos os usuários da instância. A instância seguirá os requisitos conforme Tabela 6 abaixo:

Nome da instância	Tipo	vCPUs	Memoria	Disco	SO
VM-Projeto3	Standard	1	2GB	10GB	Ubuntu 14.04

Tabela 6: Requisitos de servidor para o caso de uso

Dado os requisitos da instância, o administrador do sistema acessa a página principal do *Horizon* e realiza o *upload* de uma imagem de sistema, a colocou como pública, no caso do Ubuntu 14.04, conforme Figura 10. A visibilidade da imagem foi colocada como pública pois a mesma está localizada no repositório do administrador.

Como os usuários regulares não têm acesso aos repositórios do administrador para que a instância seja criada no projeto do usuário destino, ela deve ser colocada como pública. Outro ponto é que esta imagem poderá ser reutilizada em outros projetos, não sendo particular a um projeto. Após *upload* desta imagem, o projeto "ProjetoExemplo2" foi criado conforme Figura 11:

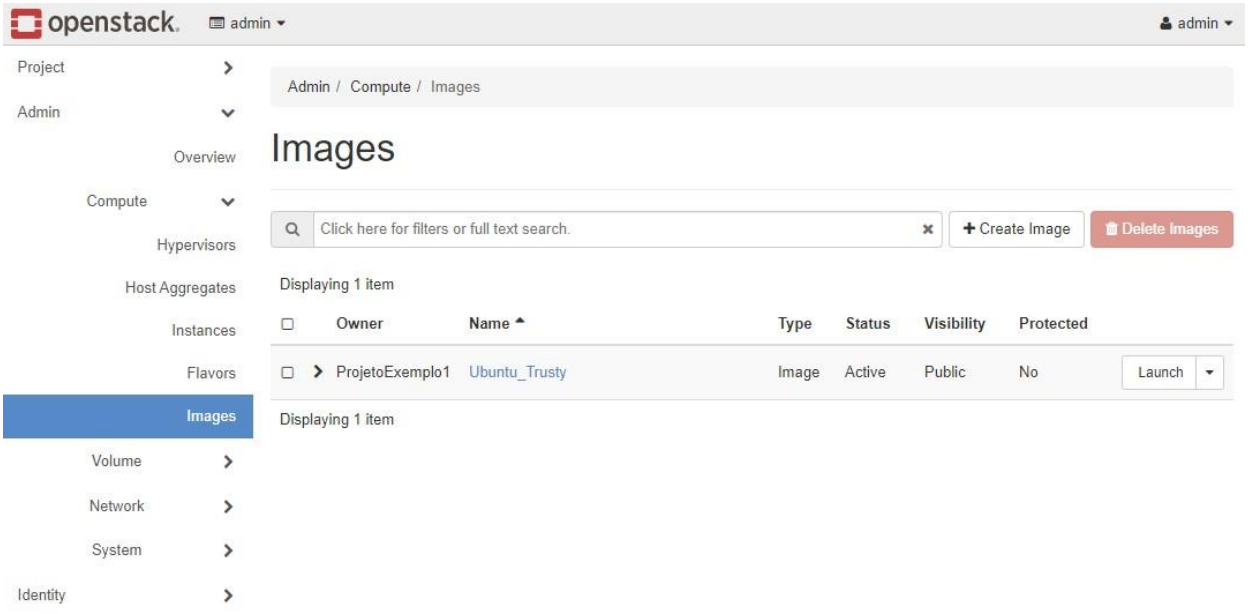


Figura 10: Imagem de sistema utilizada para provisionamento de instância

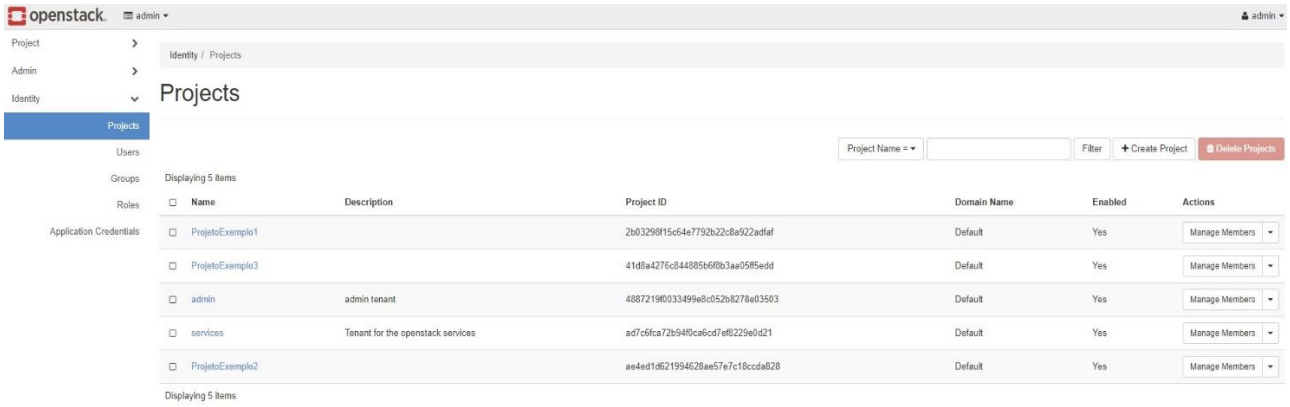


Figura 11: Relação de projetos existentes no ambiente

Após criação do projeto, o administrador deve criar uma quota, que são regras que vão limitar a criação e a utilização de recursos da nuvem. Essa é uma *feature* importante pois permite que o gerenciamento dos recursos disponíveis seja limitado a projeto ou a usuário em forma de solicitação. Ou seja, caso a solicitação de um usuário ultrapasse a cota configurada a requisição é automaticamente rejeitada. Caso contrário, o recurso solicitado pelo usuário no momento da criação será reservado e em seguida convertido em uso, caso a solicitação seja bem-sucedida.

Para o projeto em questão as cotas foram configuradas conforme Figuras 12 e 13 que mostram a relação de usuários na plataforma, incluindo o criado em laboratório para o caso de uso, Aluno 2

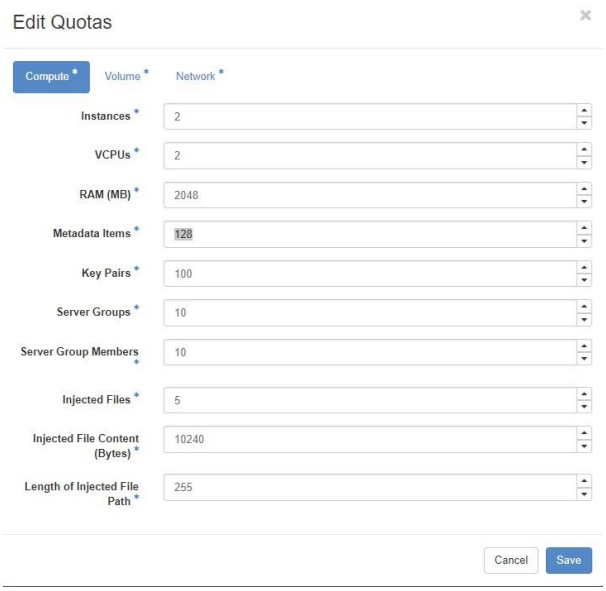


Figura 12: Critérios de cota de uso de recursos

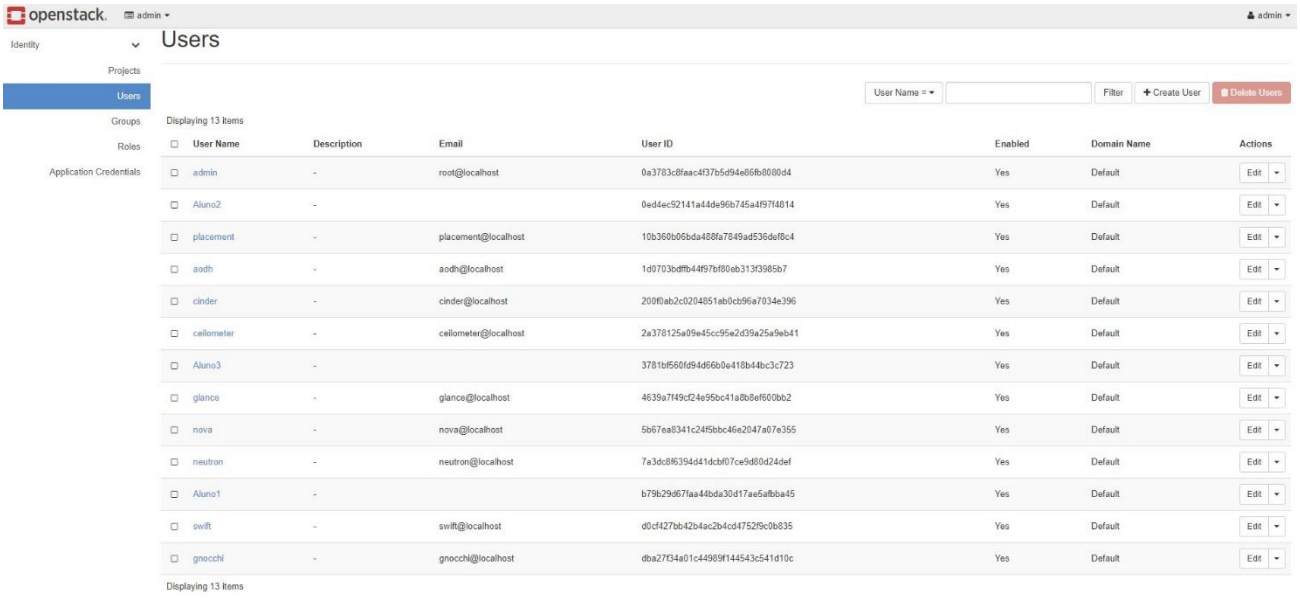


Figura 13: Relação de usuários da plataforma

O próximo passo, após as configurações de cota, é a definição ou escolha de um *flavor* para a instância. O *flavor*, conforme definido anteriormente representa o *template* de requisitos que uma instância deve possuir ao ser criada. Isto é, quanto de recurso uma instância vai consumir dos recursos disponíveis.

Essa função é importante do ponto de vista de governança pois possibilita que o administrador determine diferentes critérios dado as características dos servidores solicitados, ou dos serviços existentes no ambiente. Por exemplo, é possível ter *flavors* para cada serviço disponível do NTI, com objetivo de tornar mais rápida a criação e o *deploy* de mais instâncias com as mesmas características. Isto vale também para características de projetos, ou seja, pode-se ter um *flavor* para atender a projetos *web*, de pesquisa, de banco de dados, de rede, etc. Para este projeto, foi definido o *flavor m1.standard* que conta com especificações de instância mais simples. A mesma foi selecionada por conta de questões da infraestrutura do laboratório que não suporta instâncias com muito recurso. A relação de *flavors* disponível por padrão, em adição ao nosso *flavor* personalizado, é mostrado a Figura 14.

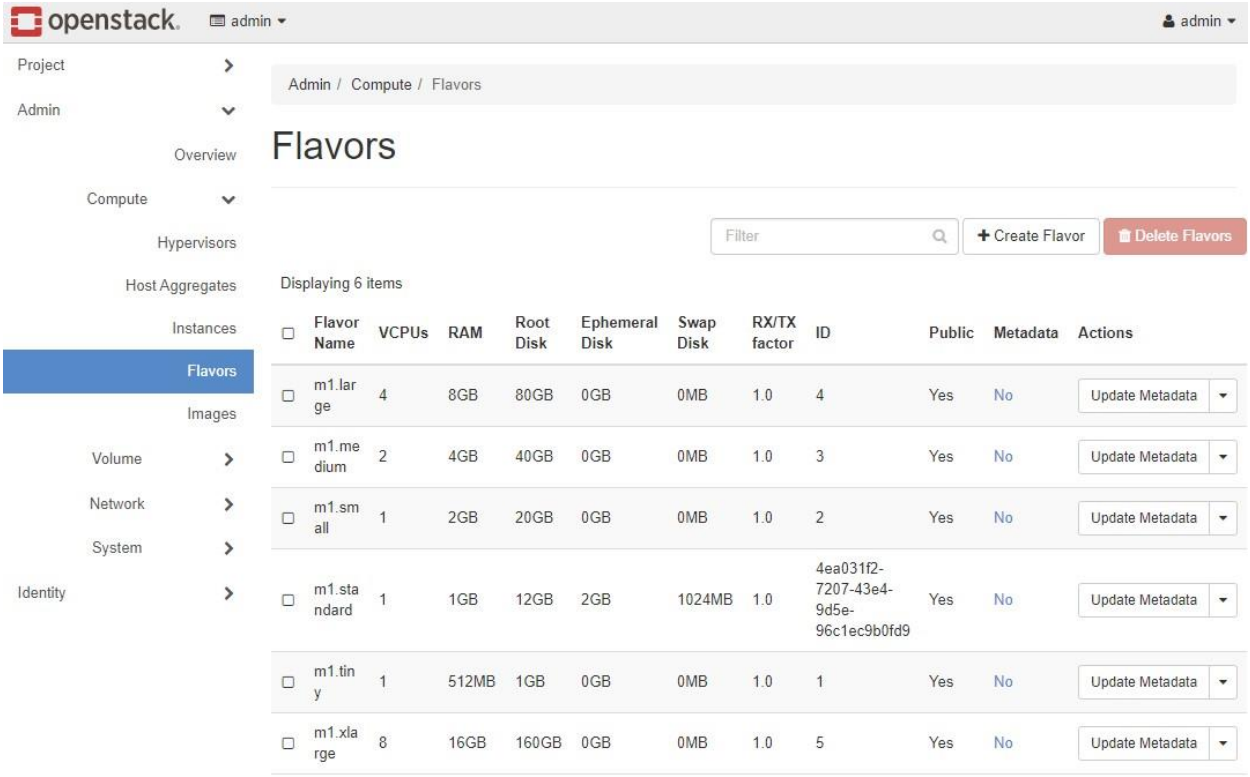


Figura 14: Relação de flavors da plataforma

O próximo passo neste momento é configurar um *security group* (grupo de segurança). Ou seja, determinar qual o tipo de acesso deve ser permitido a nível de rede, ou seja, protocolo, porta e *range* de IP que uma instância daquele projeto pode ter. Para a demanda do caso de uso, foi permitido o acesso a todos os protocolos. A relação de políticas para o *security group Standard* é mostrada na Figura 15.

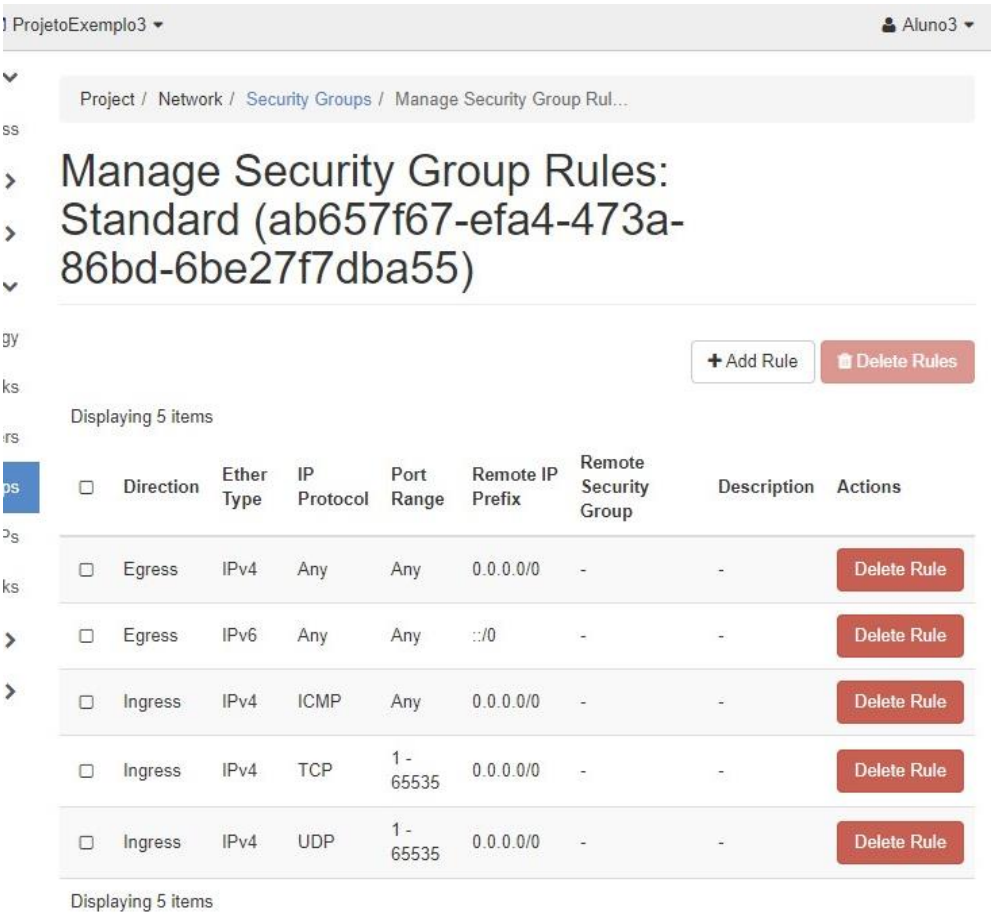


Figura 15: Relação de políticas de um *security group*

A partir deste ponto, o ambiente está com as políticas de segurança preparadas para acesso. Entretanto, caso a instância fosse provisionada neste momento, os usuários da escola não teriam acesso para realização do projeto, isto porque não foi configurado os recursos de rede para o projeto, isto é, rede de gerência e de provedor.

Sendo assim serão configurados os acessos à rede, no que tange a criação da rede de gerência do projeto e rede de provedor que será acessada pelos usuários do ambiente. As mesmas foram configuradas obedecendo o *range* de IP da rede em que o ambiente está inserido. O resultado é apresentado na Figura 16.

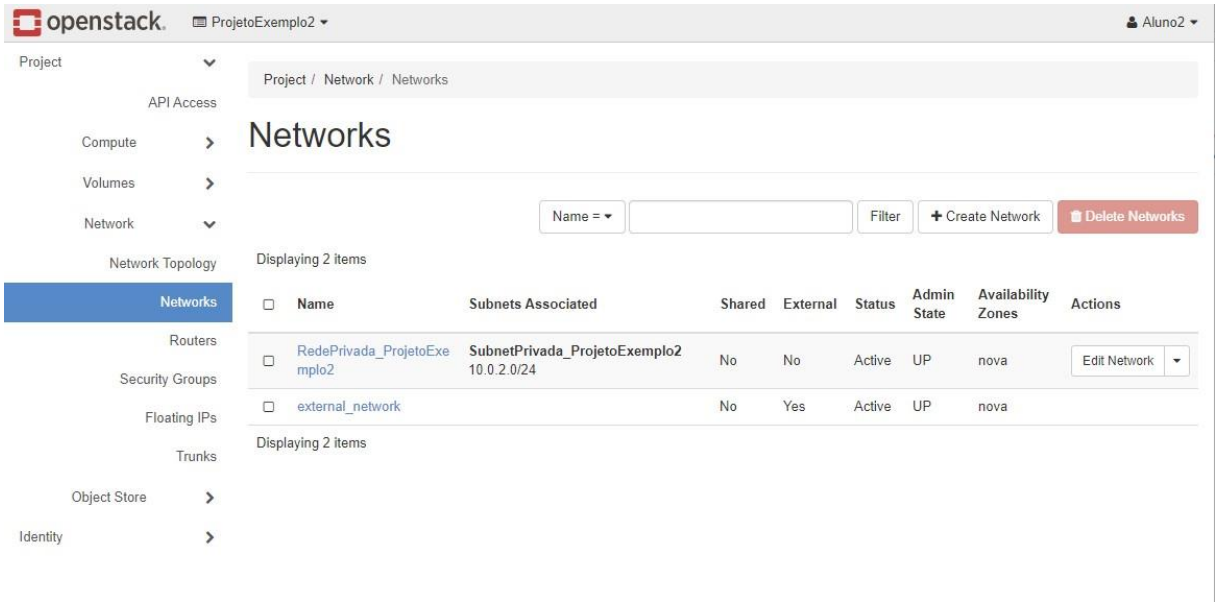


Figura 16: Relação de políticas de um *security group*

Outro recurso que deve ser criado diz respeito à configuração do IP flutuante. Ele é importante pois permite que o usuário acesse o ambiente através de um IP público, ao contrário do IP de gerência que é privado onde só se acessa uma instância partindo de outra na mesma rede privada. Além disso, um roteador virtual será configurado com o objetivo de estabelecer comunicação entre as redes recém-criadas. As definições destes dois recursos foram realizadas conforme Figuras 17 e 18. A visão da topologia de rede mostrando estes dois ambientes interligados está na Figura 19.

Figura 17: Configuração de IP Flutuante

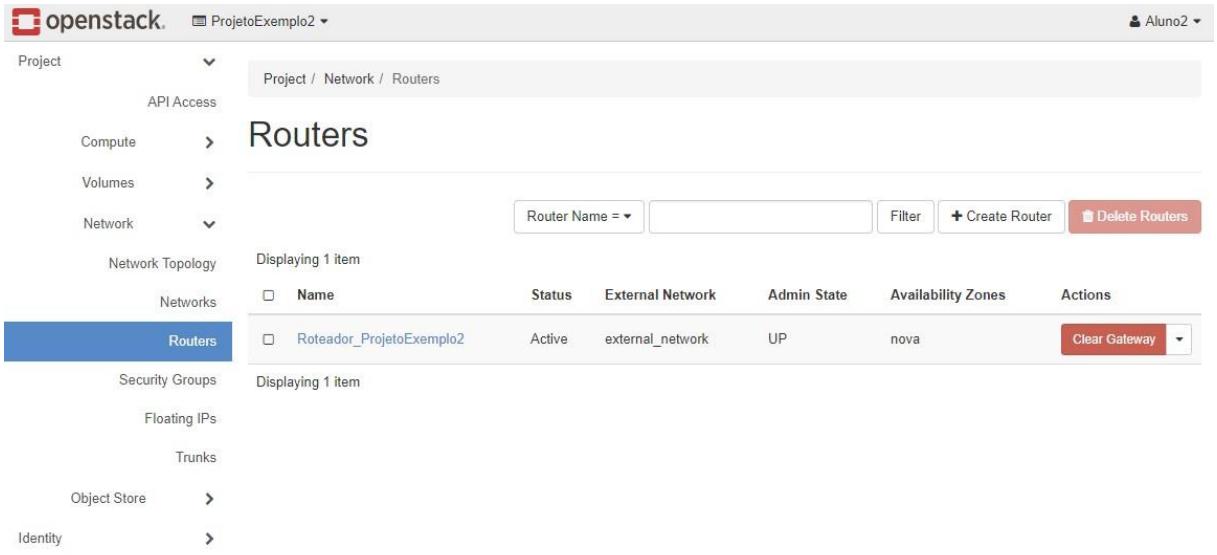


Figura 18: Configuração de roteador virtual

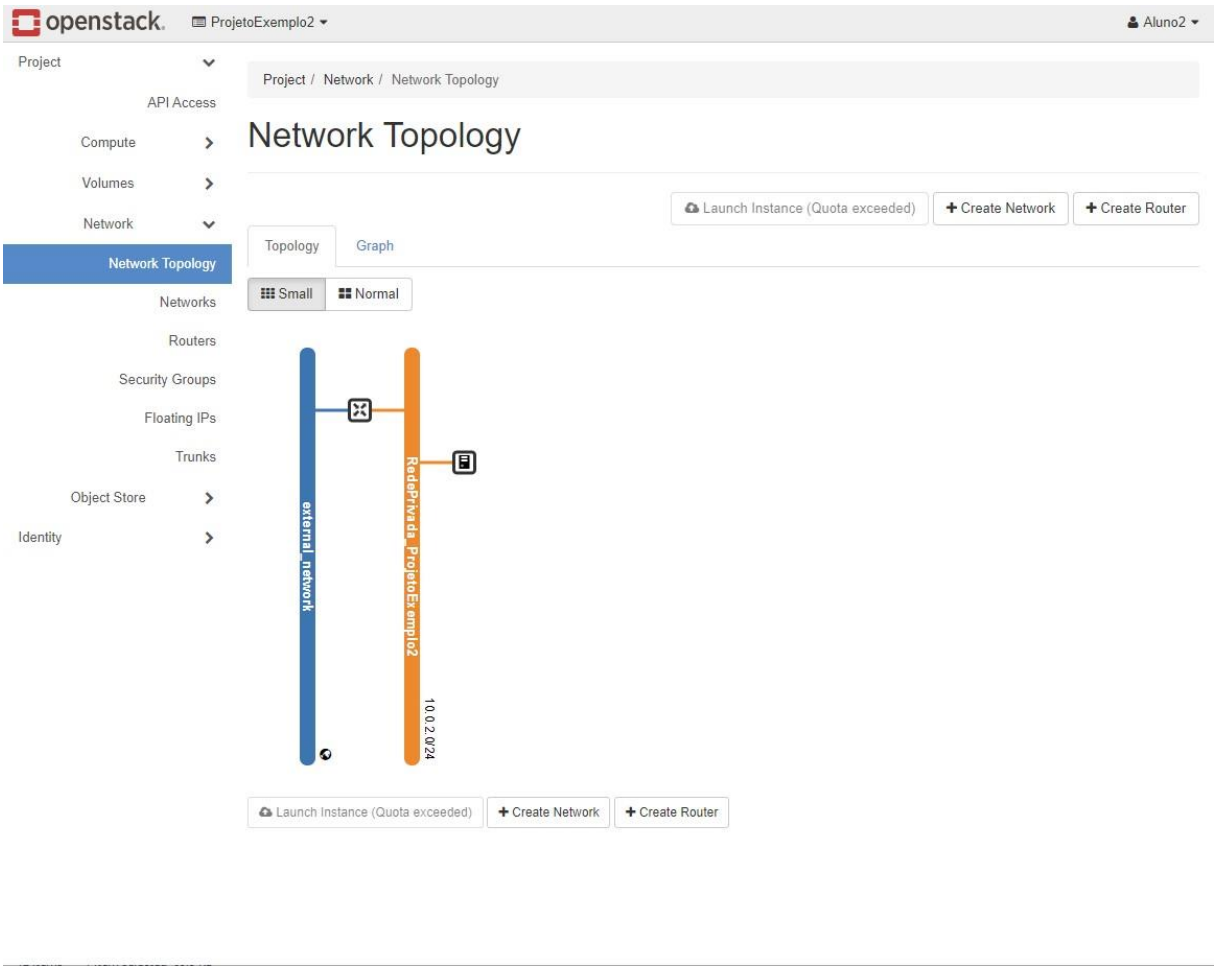


Figura 19: Topologia de rede do estudo de caso

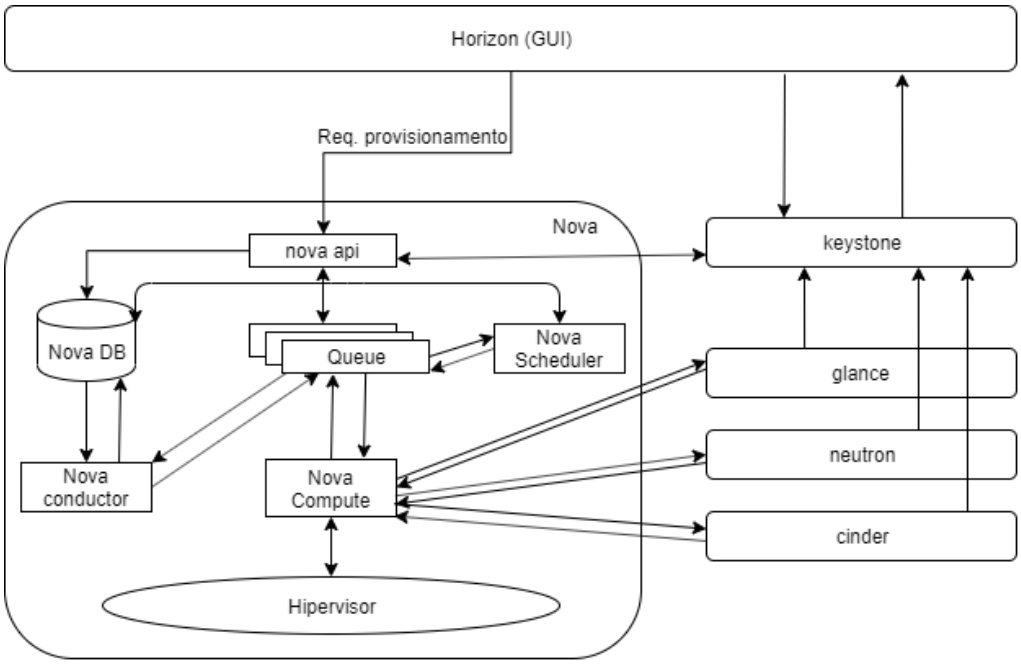
Com todas as definições de rede, segurança, políticas de usuário e *flavor*, o próximo passo foi a realização do provisionamento da instância. Quando iniciado o processo, a requisição de criação é imediatamente delegada ao módulo *nova*, que executa um procedimento de provisionamento conforme topologia representado pela figura 20, sendo realizado via REST API.

Basicamente o que o módulo faz é receber uma requisição de provisionamento com origem da interface *web* ou então da linha de comando, coletar informações de credenciais do serviço de identificação do *keystone* a fim de autenticar o usuário que está requisitando uma chamada de provisionamento, recebendo um *token* de autenticação.

Após o processo de coleta do *token*, a requisição é enviada ao nova-api que vai validar junto ao keystone o *token* gerado. Sendo esta primeira fase bem-sucedida, o *nova-api* faz uma inserção inicial no banco de dados do Nova para a instância, a partir daí o *nova-scheduler* se encarrega de coletar esses dados recém inseridos do banco e atualizar as informações de id da instância, bem como selecionar o servidor físico correto para o *deploy* da instância.

A seguir, *nova-compute* coleta informações de *flavor*, CPU e disco e se conecta passando o *token* como parâmetro via REST ao módulo *glance* para coletar uma imagem de sistema que será utilizada para o provisionamento. Após validação, nova-compute realiza da mesma forma outra chamada REST, desta vez com o módulo neutron, para alocar e configurar a rede da instância.

Por fim, o *nova* se conecta ao módulo *cinder* para coleta de informações de *block storage*, podendo opcionalmente em seguida também coletar informações de volume, se conectando logo em seguida ao *driver* do hipervisor via api ou utilizando biblioteca como a *libvirt* para a criação da máquina virtual, finalizando o processo. A partir de agora a instância pode ser vista na interface *web.*



Fonte: Blog linuxtechi[3]

Figura 20: Diagrama de provisionamento de instância

A representação da máquina virtual criada está na Figura 21, apresentando não somente a instância criada, mas o acesso a ela por parte de um usuário Aluno, isolado de forma que somente ele tem acesso aquela instância específica. Além disso, pode-se observar que os *ip's* foram alocados corretamente, juntamente com o *flavor* e a imagem, demonstrando que o provisionamento ocorreu com sucesso.

Depois deste processo, o ambiente ficou pronto para ser acessado externamente pelos usuários. O que foi feito em seguida foi a instalação do *web-server* Apache, cujo objetivo é hospedar aplicações *web* sendo realizado teste de acesso apenas pela rede da universidade se dá conforme a Figura 22.

3 Disponível em: <https://www.linuxtechi.com/step-by-step-instance-creation-flow-in-openstack/ > Acesso em: Dezembro de. 2019.

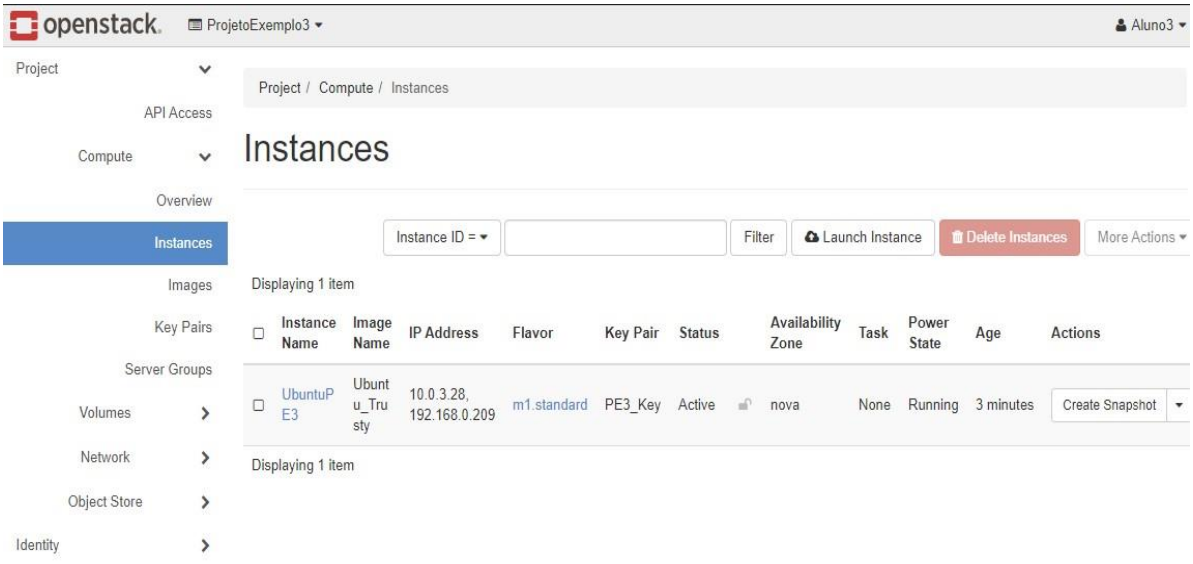


Figura 21: Representação da instância do caso de uso

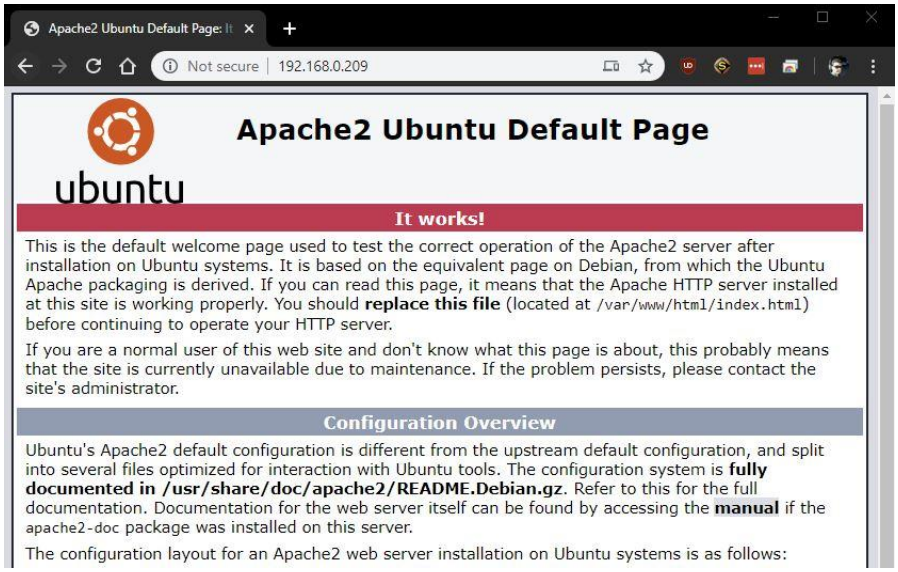


Figura 22: Representação de um deploy de um web-server sendo consumido de uma instância

Após a validação dos critérios e requisitos do estudo de caso, a demanda seria fechada pelo administrador no sistema de atendimento. Depois, o papel do administrador do ambiente é o de monitorar o consumo dos recursos deste servidor e avaliar a disponibilidade de quotas.

5 CONCLUSÕES

Este trabalho apresentou um estudo de caso sobre a implementação de tecnologia de computação em nuvem no CCET. O Capítulo 2 discutiu as definições e os benefícios do uso dessa tecnologia. O Capítulo 3 descreveu a ferramenta de gerenciamento de nuvem OpenStack. E o Capítulo 4 apresentou o estudo de caso.

O desenvolvimento deste estudo possibilitou entender os funcionamentos de uma arquitetura baseada em nuvem (tipos, modelos e classificações), entender a relação das aplicações com o ambiente físico além de trazer também duas propostas de arquitetura baseadas em melhorias nos processos do CCET atualmente.

Além disso, teve como objetivo discutir conceitualmente e de forma experimental os benefícios de uma plataforma de computação em nuvem no CCET, com a utilização do *hardware* já existente. Também, foi proposto o uso de outras funcionalidades do Openstack *para* oferecer serviços computacionais em forma de projetos, devidamente limitados, isolados e gerenciados por um administrador.

O fato do software usado na proposta deste trabalho ser *open source* é importante por permitir a adoção de uma solução de baixo custo. Isto significa, que o CCET não precisará investir na compra de *software* para modernizar o seu parque tecnológico, além de contar com recursos importantes de gerenciamento discutidos ao longo do trabalho, que são fundamentais em uma arquitetura de nuvem. Outro ponto importante remete ao fato de que apesar do trabalho ter se baseado na infraestrutura real do NTI, um maior detalhamento seria necessário para melhor aproveitamento de recursos e experimentações em temas específicos, como segurança, que não foram abordados a fundo nesse trabalho e que são determinantes na implementação em meio acadêmico.

Apesar desses problemas arquiteturais, ainda assim foram propostos dois modelos de infraestrutura ainda baseados em recursos disponíveis no NTI que podem servir como base para uma arquitetura definitiva, além de terem sido mostradas funcionalidades que de fato proporcionam mais possibilidades com os mesmos recursos.

Tendo em vista o desenvolvimento do estudo, desde a explicação de conceitos básicos, proposta de arquitetura, experimentação prática das tecnologias necessárias e suas possibilidades, pode-se concluir que o objetivo traçado no capítulo introdutório pode ser alcançado em um ambiente real.

Como trabalhos futuros, poderão ser considerados estudos teóricos aprofundados de módulos complexos do *openstack,* como o *nêutron*, pode ser discutida a interação do funcionamento do openstack com outros produtos afim de mostrar a interoperabilidade com outros sistemas que podem complementar o seu uso como o *VSphere*, da *VMWare*, podem ser realizados estudos voltados para segurança em computação em Nuvem com o objetivo de diminuir a vulnerabilidade de dados sensíveis, como os dados acadêmicos, além de estudos voltados para projeto e gestão de ambientes de Nuvem.

Referências Bibliográficas

1. Qi Zhang · Lu Cheng · Raouf Boutaba (2010) “Cloud computing: state-of-the-art and research challenges”. The Brazilian Computer Society 2010. 2
2. Vaquero L, Rodero-Merino L, Caceres J, Lindner M (2009) “A break in the clouds: towards a cloud definition”. ACM SIGCOMM computer communications review
3. Yashpalsinh Jadeja, Kirit Modi (2012) “Cloud Computing - Concepts, Architecture and Challenges”. 2012 International Conference on Computing, Electronics and Electrical Technologies [ICCEET]
4. Kuyoro S. O., Ibikunle F. & Awodele O. (2011) “Cloud Computing Security Issues and Challenges”. International Journal of Computer Networks (IJCN), Volume (3)
5. Relatório de Gestão do CCET Exercício 2017 - “http://wwwp.uniriotec.br/ccet/wp-content/uploads/sites/39/2019/05/Relat%C3 %B3rio-de-Gest%C3%A3o-do-CCET-exerc%C3%ADcio-2017.pdf” [Acessado em Setembro, 2019].
6. Ben Silverman,Michel Solberg - OpenStack for Architects 1st Edition - Packt Publishing (3 de fevereiro de 2017)
7. www.openstack.org.[Acessado em Setembro 2019]
8. https://aws.amazon.com/pt/ [Acessado em Novembro, 2019].
9. https://cloud.vmware.com/ [Acessado em Novembro, 2019]
10. https://www.ibm.com/br-pt/cloud [Acessado em Novembro, 2019].
11. https://docs.openstack.org/arch-design/arch-requirements/arch-requirements-ha.html [Acessado em Novembro,2019]
12. https://docs.openstack.org/ha-guide/intro-ha-key-concepts.html [Acessado em Novembro, 2019]
13. https://docs.openstack.org/ha-guide/intro-ha-common-tech.html [Acessado em Novembro, 2019]
14. http://www.haproxy.org/ [Acessado em Novembro, 2019]
15. SILVERMAN, BEN; SOLBERG,MICHAEL. OpenStack For Architects; 1. Ed.Editora OReilly,2017
16. https://docs.openstack.org/liberty/networking-guide/intro-os-networking-overview.html [Acessado em Dezembro, 2019]
17. https://docs.openstack.org/arch-design/design-control-plane.html [Acessado em Dezembro, 2019]

18. SEFRAOUI, O, AISSAOUI, M., ELEULDJ, M. (2012) “OpenStack: Toward an Open-Source Solution for Cloud Computing”. International Journal of Computer Applications (0975 - 8887) Volume 55 - No. 03, October 2012

19. https://docs.openstack.org/ocata/networking-guide/deploy-ovs-selfservice.html [Acessado em Dezembro, 2019]

20. https://docs.openstack.org/python-openstackclient/pike/cli/command-objects/keypair.html [Acessado em Dezembro, 2019]

21. https://www.linuxtechi.com/step-by-step-instance-creation-flow-in-openstack/ [Acessado em Dezembro, 2019]

22. https://www.redhat.com/en/topics/cloud-computing/what-is-cloud-management [Acessado em Dezembro, 2019]

23. https://www.eucalyptus.cloud/ [Acessado em Dezembro, 2019]

24. https://docs.openstack.org/arch-design/arch-requirements/arch-requirements-enterprise.html.[Acessado em Novembro de 2019]

25. https://docs.openstack.org/arch-design/arch-requirements/arch-requirements-enterprise.html [Acessado em Novembro de 2019]

26. https://docs.openstack.org/glance/latest/?_ga=2.232109596.2098271895.1572475554-1534350400.1561136582 [Acessado em Outubro De 2019]

27. https://docs.openstack.org/keystone/latest/admin/identity-concepts.html [Acessado em Outubro De 2019]

28. https://opennebula.org/comparing-opennebula-and-openstack-two-different-views-on-the-cloud/ [Acessado em Dezembro 2019]

29. https://docs.openstack.org/neutron/latest/admin/intro.html [Acessado em Outubro De 2019]

30. https://blog.openshift.com/what-is-platform-as-a-service-paas/ [Acessado em Dezembro De 2019]

31. https://gsuite.google.com/features/ [Acessado em Dezembro De 2019]

32. https://www.openstack.org/software/releases/train/components/horizon [Acessado em Dezembro De 2019]

33. https://docs.openstack.org/swift/latest/admin/objectstorage-intro.html [Acessado em Dezembro De 2019]

34. https://docs.openstack.org/stein/ [Acessado em Dezembro de 2019]